EMPOWERING USERS TO COMMUNICATE THEIR PREFERENCES TO MACHINE LEARNING MODELS IN VISUAL ANALYTICS

A Dissertation
Presented to
The Academic Faculty

by

Subhajit Das

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
May 2021

EMPOWERING USERS TO COMMUNICATE THEIR PREFERENCES TO MACHINE LEARNING MODELS IN VISUAL ANALYTICS

Approved by:

Assistant Professor Dr. Alex Endert,
Advisor
School of Interactive Computing
Georgia Institute of Technology

Associate Professor Dr. Thomas Ploetz
School of Interactive Computing
Georgia Institute of Technology

Regents Professor Dr. John Stasko
School of Interactive Computing
Georgia Institute of Technology

Associate Professor Dr. Remco Chang
School of Computer Science
Tufts University

Associate Professor Dr. Polo Chau
School of Interactive Computing
Georgia Institute of Technology

Date Approved: November 20 2020

PREFACE

This dissertation is original work by the author, Subhajit Das. Portions of this dissertation have been published, or have been submitted for review prior to publication. There have been significant contributions by co-authors for portions of this work, which have been noted in the relevant sections.

Not everything that can be counted counts, and not everything that counts can be counted.

-Albert Einstein

For MA, BABA, DADA, and Florina ....

ACKNOWLEDGEMENTS

I want to acknowledge the people who guided me throughout my journey. The completion of this work was made possible by the encouragement from my family, friends, mentors, and colleagues, who instilled courage in me in various ways. In that spirit, first, I want to thank my parents and my brother for always believing in me and pushing me to keep working hard. I also want to thank my wife Florina for standing with me throughout this journey in every step.

In my academic life, first and foremost, a big thanks to my thesis committee for being patient and encouraging at every step of the process; and for giving me the freedom to explore and find novel ways to solve problems. Next, I wish to thank my advisor, Dr. Alex Endert. Thank you so much for believing in me, even when I was struggling through research. It has been a great pleasure working with Alex. Throughout the process, I learned how to be structured, organized and how to present research in academia as an independent researcher.

Thanks to Georgia Tech's incredible Visualization Lab mates for all the feedbacks, critiques, ideas and for showing high-quality work that pushed me to think better and hard at some of the problems that are seminal parts of my thesis. Special thanks to my seniors Bahador Saket and Emily Wall for being wonderful colleagues and mentors and for setting great examples for me to follow.

Many thanks to the Darpa D3M Team, especially to Dr. Remco Chang and Dr. Michael Gleicher, for being such a great advisor and mentor. I learned so much in the last four years from all of you (D3M Team: Dylan Cashman, Shah Rukh Humayoun, Florian Heimerl, Shenyu Xu, Cong Liu, Ab Mosca, Ashley Suh, Bahador Saket, John Thompson) and made so many wonderful memories with everyone in this team.

I express my gratitude towards many faculty members whose classes at Georgia Tech nurtured my understanding of computing, visualization, and AI. In particular, I want to thank Dr. Jim Foley, Dr. John Stasko (Information Visualisation), Dr. Byron Boots (Machine Learning), Dr. Greg Turk (Simulation of Biological Systems), Dr. Jarek Rossignac (Computer Graphics), Dr. John Haymaker (Design Space Construction), and Prof. Chuck Eastman (Data Modeling for Healthcare Planning). Without them, my academic journey would be much more difficult. They provided me with the tools that I needed to successfully complete my dissertation.

I would like to acknowledge my colleagues and mentors from the numerous internship opportunities I had, including Autodesk (Matt Zeyk, Anthony Hauck, Collin Day), Bosch Research (Panpan Xu, Zeng Dai, Liu Ren), Microsoft Research (Darren Edge), and UPS Advanced Technology Group (Joan Chmielewski, Tim Major). They showed me how my skills and ideas can be applied in the real world; I thank them for taking a chance on me and for the opportunity to see things from a new perspective.

Last but not least, I would like to thank all my friends who have been supportive of my work and were kind throughout my dissertation. Finally, I thank my best friend Nilesh Maurya for the unconditional love, support, and trust in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Recent visual analytic (VA) systems rely on machine learning (ML) to allow users to perform a variety of data analytic tasks, e.g., biologists clustering genome samples, medical practitioners predicting the diagnosis for a new patient, ML practitioners tuning models' hyperparameter settings, etc. These VA systems support interactive construction of models to people (I call them power users) with a diverse set of expertise in ML; from non-experts, to intermediates, to expert ML users. Through my research, I designed and developed VA systems for power users empowering them to communicate their preferences to interactively construct machine learning models for their analytical tasks. In this process, I design algorithms to incorporate user interaction data in machine learning modeling pipelines. Specifically, I deployed and tested (e.g., task completion times, user satisfaction ratings, success rate in finding user-preferred models, model accuracies) two main interaction techniques, multi-model steering, and interactive objective functions to facilitate specification of user goals and objectives to underlying model(s) in VA. However, designing these VA systems for power users poses various challenges, such as addressing diversity in user expertise, metric selection, user modeling to automatically infer preferences, evaluating the success of these systems, etc. Through this work I contribute a set of VA systems that support interactive construction and selection of supervised and unsupervised models using tabular data. In addition, I also present results/findings from a design study of interactive ML in a specific domain with real users and real data.

CHAPTER I

INTRODUCTION

People work with data for numerous reasons such as exploratory data analysis, sense-making, predictive modeling, and tradeoff analysis for decision making, etc. For example, realtors look at real estate data to identify houses to sell or buy or to find neighborhoods where property prices may rise or fall. Various data analytics and visualization tools (e.g., Tableau, Spotfire, MS Excel, etc.) support these data analysis workflows. However, with the advent of faster machines, cheaper memory, and easier access to heterogeneous large data sources, current data analysis approaches can utilise more complex methods [118], which can empower users to explore data with a new perspective. Recent work in the field of visual analytic (VA) systems addresses this challenge by designing novel visual data analysis interfaces and interaction techniques that support advanced statistical methods such as machine learning (ML) to gain insight from the data. These systems facilitate analytical tasks such as prediction, grouping/clustering, graph matching, metric learning, labeling, etc. For example, realtors may use a VA system to interactively construct a regression model to predict the price of a house; this may help them decide on whether to buy a potential property or not. In this research I have investigated how people can interactively construct, and select machine learning models (supervised and unsupervised) using tabular data and a visual interface (without explicitly writing code).

1.1 Problem Statement

The advent of data and machine learning (ML) has impacted various domains, such as in health care, where machine learning is used to predict patient diseases or to find similar patients to prescribe diagnosis. Similarly, self-driving cars rely on advanced ML models that are very efficient to segment images in real time to detect on road obstacles such as other cars or people etc. All of these mission-critical decision-making tasks are driven by complex ML models, which people in different domains are using for various use cases. Here a ML model refers to: (1) a mapping of the data (e.g., images of animals, properties of houses, etc.), to an output or a variable in the data that people may wish to predict (e.g., labels of images such as cat or dog), (2) a low dimensional representation of the input data, or (3) a characterization of the data that tells association, or grouping between the data items. In practical applications the process of constructing these models follows a complex modeling pipeline (see Figure 1), that includes problem specification, data acquisition, data processing, model selection, model validation, and model deployment for production.

However, end-users who need ML often only take part when specifying problems or acquiring data; or they only take part when models are deployed, and they provide feedback to further improve their performance as they use it. Current workflows of model construction do not let end-users be part of the modeling phase where they can directly adjust a model's behavior when required. In comparison, ML practitioners and data scientists take part in modeling and deploying models using sophisticated software systems. However, they usually do not acquire data or might not know much about the domain in which the model is to be used. Thus, these users cannot independently process the data to build ML solutions.

They rely on domain experts for this part. This is where the problem is; people who need ML are still not active participants of this modeling process. Either they build models or acquire/provide data.

If people who are end-users (e.g., doctors, lawyers, analysts) only provide data, but are unable to: (1) adjust model(s) behavior or (2) interactively probe or construct models to verify if it learned the right characteristics of the data, they might not use it in their analysis process. For example, if a medical practitioner cannot not probe a model to verify its behavior, they may not use it to predict patient diagnosis. Similarly, if they construct a model using any of the current Auto-ML tools such as H2O, Auto-Weka, etc. and intend to improve the model in some aspects, they may have to rely on ML practitioners or learn to write code. Other potential users of interactive ML systems, can be expert ML users such as data scientists. When these users construct model(s) they may want to use a graphical user interface that not only speeds their current processes of model construction, but also allows them to explore various model options along with the training data. This can help them in many day to day model building tasks such as debugging models, exploring a number of model hyperparameterizations, exploratory data analysis for feature extraction/construction, reporting model outputs to other team members and/or end-users.

To resolve this gap, researchers and scholars have been working to incorporate research and insights from HCI and Machine Learning community, to include humans in the loop of ML modeling [99] also called human-centered ML or interactive ML [75], where users can actively take part in every aspect of modeling. Along this process, humans can bring in domain expertise to the model training process, and also help in finding the right data for the task as opposed to the conventional process of utilizing lots of data with the possibility of noise in it. Using this workflow humans can interactively build better performing models and using domain knowledge can probe models to ensure that they can be trustfully deployed in real-world applications. In essence, including humans in every stage of the modeling pipeline has many advantages, which based on the literature review of current VA systems, seems to require further research.

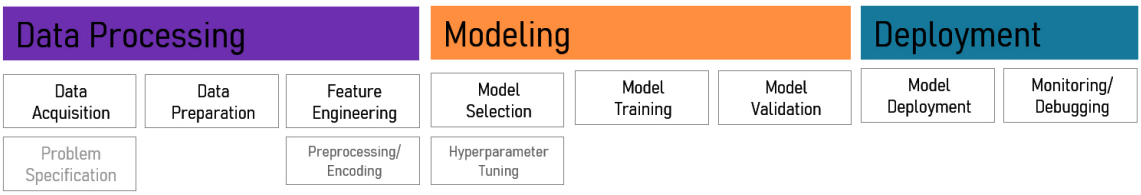


Figure 1: A typical ML modeling pipeline that is used to construct and deploy ML models in practical applications.

1.2 Vision: Machine Learning for Power Users

In this research I am motivated to include humans in the loop of ML modeling. Usually, THE process of model construction is iterative (see Figure 2), where per iteration, people test

Table 1: Study by Yang et al. interviewed ML users from various professions, who use ML-based processes to solve a diverse set of problems [248].

Profession	Example ML Problem
Professional Software Engineer	Bug Report classifier
Project Manager	User Feedback Classifier
Manager	HR Policy QA bot
Business Analytics	Predictive Machine Maintainence
Artist	Emotion Classifier for wearables Machine
Botanist	Predictive plant nutrient maintenance
Academic Researcher	Sensor signal classifier maintenance
Clinical Researcher	Prognostic classifier classifier maintenance
Mechanical Engineer	Insurance Risk Estimate

different hypotheses, model types, and hyperparameter settings to improve model performance (such as accuracy, or other defined performance parameter). Typically this workflow includes programming/coding by experienced ML practitioners or experts; however, when this process is made interactive, e.g., using graphical user interface (GUI) widgets, end-users such as analysts can be active participants to adjust/steer and select models suited their goals. While existing VA systems support ML-based processes for various data analysis goals, interactive construction, selection, and steering of ML models for personalized user goals is under-explored; it needs further research to make ML accessible for various use cases and problem domains.

Motivated to make ML modeling pipelines interactive, first through prototyping a set of VA systems, I sought to empower people to specify input (through interaction with a visual interface, without writing code) to ML problems such as provide labels, weight data instances, specify relevant features, etc. Along the process, I designed algorithms that model end-users behavior based on choices they make as they interact with such VA systems. In this workflow, user interactions were logged and further analyzed in real time to provide personalized model recommendations; by inferring or predicting their choices with the goal to empower them to select or interactively construct models.

However, as good as it sounds, I was curious to investigate who are these end-users. Who are these people who need to be empowered to specify their preferences to ML models (interactively)? Based on the literature review (Chapter 2), people who need ML include ML practitioners who validate models by interactively tuning their hyperparameters, medical practitioners interactively constructing classifiers to predict a suitable diagnosis for patients, hobbyist artists who construct models to deploy creating art projects using technology, data scientists interacting with Auto-ML to test various model hypothesis, and so forth. Along the same lines, Yang et al. conducted a survey study of 100 such users (e.g., software

engineers, artists, botanists, etc., see Table 1) who use machine learning for various tasks [248].

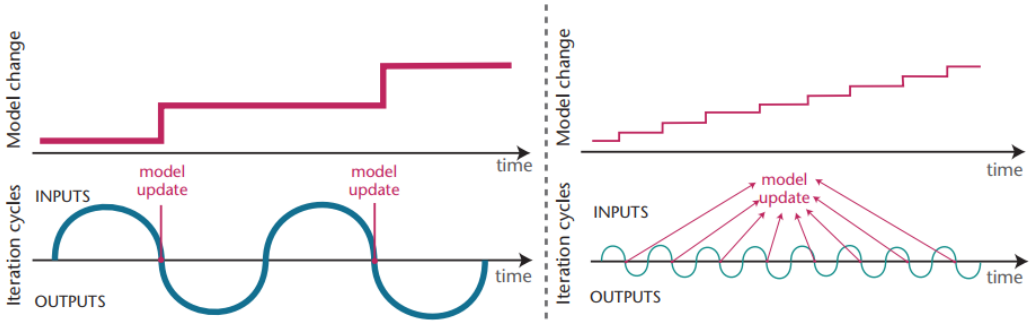


Figure 2: The top row shows a traditional machine learning process, which is linear. The bottom row shows an interactive machine learning process that includes interactions as input, then model update, then change in output. The process is iterated until the user is satisfied with the output. [10]

These users construct ML models from various sources such as, ready to use scripts/codes from the web, Auto-ML tools such as Auto Weka [82], consulting with ML practitioners, utilising self-written codes, etc. In this research, VA systems that I build are designed for these users who I also call "power users".

Power users exhibit a wide spectrum in their expertise or skillsets in ML. While some are ML practitioners/experts, referred to as *expert ML users*, others are domain experts or end-users or analysts, who need ML for analytical tasks, but do not know how to code ML applications. These people utilise ML as a black-box tool to support their goals as *non-experts in ML* as referred here [248]. Within these two extremes, there are *intermediate* users who know programming/coding but do not theoretically know ML (e.g., software developer users of the system Gestalt [168]). My research explores interactive techniques to design systems that support people to interactively construct ML models for diverse set of use cases.

My research supports power users in communicating their preferences (data analysis goals) to ML models, and empowers them to interactively construct, select, and steer (adjust) these models using VA systems. To that end, I investigate numerous interactive VA techniques such as model selection, multi-model steering, interactive objective functions, and conflict resolution in objective specifications (see Section 1.3). Through this research, I explain each of these techniques using prototype VA systems that are further deployed and tested with real users (see Figure 4 for the systems I built that support various user tasks). I validated the effectiveness of these systems by collecting and analysing data to measure task completion times, user satisfaction ratings, success rates of finding a user-preferred model(s), model accuracies along with qualitative user feedback about the system's usability. I describe lessons learned from the evaluation of these systems that further enriched my understanding of how power users communicate their preferences to ML models through the designed interactive techniques.

1.3 Terminologies

In the following, I define a set of terminologies seminal to the contributions of this thesis, and to ensure readability of the rest of the document.

Model: A model is a mapping from an input space to an output space. In the context of machine learning, an input space is the input data (i.e., images of animals), while an output space is a variable in the data that users wish to predict (i.e., labels of images such as cat or dog) or a low dimensional representation of the input data. Mathematically, a *learning algorithm* A maps a training set D_{train} to a model f by searching through a parameter space. A model is described by its *parameters* θ, while a learning algorithm is described by its *hyper-parameters* λ. A model parameter is internal to a model, where its value can be estimated from the data. Model parameters refer to values that are learned during the training of a model, such as coefficients, while hyperparameters are typically determined through some process external to training, such as cross-validation.

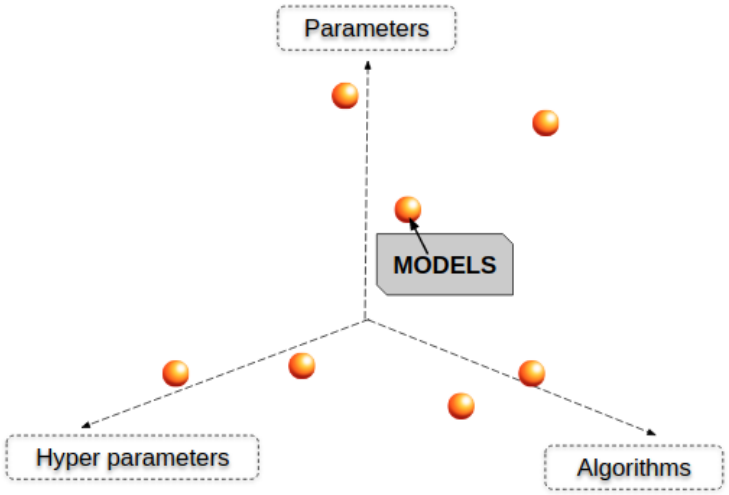


Figure 3: Simplified view of a hyper-dimensional model space, showing three of its dimensions. Orange spheres represent different models in this view.

Model Space: A model is constructed by a careful selection of a learning algorithm and associated hyperparameters. VA systems adjust the underlying model(s) by dynamically changing it's hyperparameters or choosing a new learning algorithm. Various combination of learning algorithms and hyperparameters give rise to a vast number of different model types. These different models constitute an exhaustive high dimensional model space from which various models can be sampled using a unique combination of a learning algorithm, and its associated hyperparameters. For example, a support vector machine model uses a *poly* kernel function with other hyperparameters such as *C-value*, *gamma*, etc. I define this space of different model types as *model space*. This also goes along with the definition of model space defined by Eli et al. [30]. In this high dimensional model space, a model is plotted as a point (see the gray box in Figure 5). It is noteworthy to emphasize that this model space is hyperdimensional with densely plotted models, owing to a plethora of continuous and discrete hyperparameter variables (see Figure 3).

	What user does -	What system infers and applies -				
		Multi-Model Steering			Interactive Objective Functions	
	USER OPERATIONS/TASKS	BEAMES	Gaggle	Geono-Cluster	QUESTO	CACTUS
DATA	Show data similarity by dragging visual data marks e.g., circles etc.	-	Trains a binary classifier and learns a ranking function	Learns cluster distance function	-	-
	Select a single visual data mark such as circles	Adjusts data item weight	Specifies labelled training data	Specifies cluster membership	Specifies data items' weight and similarity	-
	Select multiple data marks by lasso operation	-	-	Specifies similarity to multiple data items	-	-
	Inspect model output on test/validation data set	Uses a new validation set per iteration	-	-	Uses a new validation set per iteration	Uses a new validation set per iteration
FEATURES	Specify feature weights	Specifies feature weights as prior to least squares function	-	Specifies normalized weights to similarity distance function	-	-
	Select relevant features	Discards redundant features	-	Discards redundant features	Discards redundant features	-
	Specify variance and correlation	-	-	-	More variant & less correlated features are highly weighted	-
	Inspect feature contributions	Shows a bar chart of feature weights	-	Shows a bar chart of feature weights	Shows names of top 3 most important features	-
MODELS	Build models seperately from the GUI that presents model outputs	-	-	-	-	Define models in Python separately
	Inspect multiple model outputs	Shows n = 100-200, regression models	-	Shows output of n = 4 recommended clustering models	Shows top n = 5 classifiers	Shows one model per iteration
	Inspect models over time (incremental model construction)	Saves/exports models per iteration	-	-	-	Shows one model per iteration
	Compare models by output	Visualises multiple models by color and text side by side	-	-	Visualises multiple models by color and text side by side	Visualises models by spark bar charts and text
	Build ensemble models	Builds ensembles from user-specified models	-	-	-	-
OBJECTIVES	Specify objective weights	-	-	-	Translates specified weights to normalised weights to the function	Translates specified weights to normalised weights to the function
	Inspect objectives specified as preferences	-	-	-	Visually shows objective weights	Visually shows objective weights
	View and export conflicts found in specified objectives	-	-	-	-	Exports conflicts detected to a Jupyter notebook
	Resolve conflicts	-	-	-	-	Removes or re-assigns user specified examples to objectives
	Compare many objective functions	-	-	-	-	Saves and shows objective func. per time step for exploration

Figure 4: Summarizes VA systems that I have developed as part of this research, and various user tasks it supports. The rows lists user tasks and the columns lists implication on the system.

Furthermore, in the model space, power users may search for models based on their complexity or performance in terms of compute time. For example, power users may prefer a model that finishes its task in an hour as opposed to a model that takes months to run at the cost of some loss in its accuracy. However, conventionally an expert ML practitioner might select a more accurate model even if it takes significantly longer compute time. Thus the definition of a desired or optimal model (from the model space) may be different based on the needs/goals of the user.

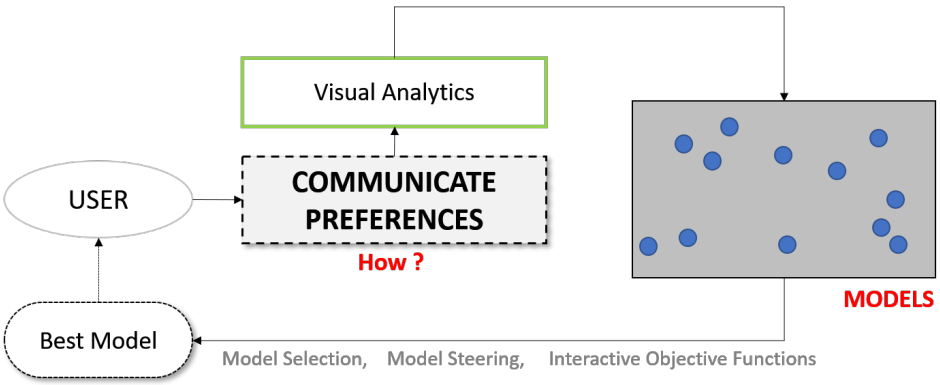


Figure 5: Summary of my thesis work explaining my primary research question: How do power users communicate their preferences to models?

Model Selection: Different types of machine learning models can be constructed for the same data, problem, and task. For example, a classification task on a tabular data can be supported by a Decision Tree, Random Forest, or a Support Vector Machine (SVM) model. This requires making an informed decision to select the right model for the task and the data. Selecting a model for the said task is called *model selection*. Inherently, model selection is a complex process conventionally done by experienced ML practitioners. Model selection requires critical evaluation of models, iterative exploration of their performance or behavior, and understanding the problem domain correctly.

Model steering: When a model is selected, users can inspect its performance by reviewing defined metrics such as cross-validation score or by reviewing the model's output, such as predicted labels on an input image data. If users are not satisfied with the output, they may need to refine the performance of the model. Conventionally, experienced ML practitioners make such adjustments by changing the model's learning algorithm, hyperparameters, or by defining a different cost/objective function to update the underlying parameters. However, in VA systems end-users can adjust model performance by directly interacting with the visual data representations/encodings, also known as direct manipulation based interaction (see Chapter 2). For example, in the visual analytic system Dis-function users can drag two circles in a scatterplot closer to specify their similarity to each other [29]. In response, the system updates model parameters to achieve the specified goal (see Figure 6). This approach to adjust a model by interacting with visual data representations in an interface is called *model steering*. However there are various ways model steering can be deployed in a VA system. For instance, a VA system embedded with a predefined single ML model exhibits single-model steering. On the other hand, recently there are VA systems which facilitate construction of multiple ML models to satisfy user objectives such as Clustervision, Hypermoval, etc. [131, 179]. These systems, in response to user interactions, update model hyperparameters of multiple ML models simultaneously to automatically adjust their behavior. This approach is called multi-model steering.

Model selection and multi-model steering are very similar yet distinct concepts. While model selection investigates finding the right model for the data, task, and the problem domain, model steering facilitates improving/adjusting model behavior by updating its hyperparameters and parameters. However, both of these techniques help users to find the right (optimal) model based on the choices they specify. In addition, both model selection

and model steering help users interactively navigate the model space in search of regions where the likelihood of finding better models is higher. In essence, these techniques deploy statistical methods based on specified user interactions triggering smarter searches in the densely plotted high dimensional model space (see Figure 5).

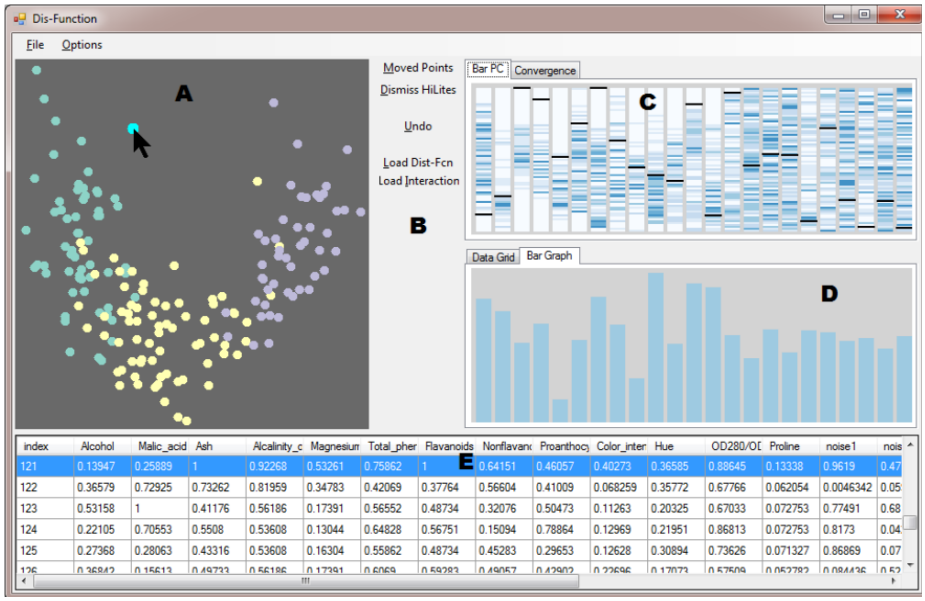


Figure 6: The interface of Dis-function, a VA system that enables users to change the underlying function in a model. Users can specify similarity or dissimilarity between data items by dragging points in the scatterplot on the left. [29]

Interactive Objective Functions: While multi-model steering was effective, it seemed from the interactions in the other systems, that users can specify a range of implicit feedbacks that often are not communicated back to the user. In some cases, users may not understand how each of these preferences affect each other, and more importantly often these preferences when applied together may conflict with one another. Every ML model uses an objective function, often called loss or cost function. Through my research, I prototyped a novel VA technique called interactive construction of objective functions. This technique visualizes an objective function showing it's sub-objectives/component objectives, and constraints. Furthermore, it allows users to interactively specify these sub-objectives to suit desired goals. Interactive construction of objective functions allows modeling ML solutions that cater to specific user requirements. Visual representation of the objective function explicitly show users the set of constraints or sub-objectives that are satisfied or vice versa. The interactive objective function may also reveal conflicting sub-objectives which may present to users the need to review their preferences further. To my knowledge, this technique is not practiced, applied, or deployed in any existing VA system in the literature of visual analytics. Through this work, I showed interactive objective functions empowers people to better satisfy personalised goals (using Auto-ML model solvers) as opposed to relying on stand-alone Auto-ML workflows.

1.4 Current Approaches in Visual Analytics

Over the years, VA systems have helped users glean insights and discover hidden patterns in data. These systems have shown to handle massive, heterogeneous, and dynamic sources of data by presenting visual data representations (data items encoded as visual graphical user elements) integrated with user interactions. They facilitate users to process and analyze vast amount of information fluidly [117] by facilitating visual data analysis. Further, these visual data representations help reduce users cognitive load to process massive information from dynamic and heterogeneous data sources. VA systems support a wide array of tasks ranging from high to low such as, data exploration, data comparison, ranking, summarization, storytelling, making sense of large data, etc. [67, 87, 90, 151, 219, 221]. Extending this list of data analysis tasks, recent VA systems have integrated ML-based methods. ML in VA gave rise to interactive machine learning (IML) systems [76] which allows advanced data analysis tasks such as, classification, regression, clustering, graph matching, etc. [10, 106, 188]. For example, using a system like Hypermoval [179] users can construct and review multiple regression models. Past efforts in integrating ML models with VA systems visually represented model outputs as graphical encodings, just like visually encoding data values. For example, researchers visualized decision boundaries by plotting data points as dots in a scatterplot, and encoding decision surfaces as a line separating the dots [80, 191]. This method can be easily used to represent output from a diverse set of ML models (e.g., SVM, Decision Trees, Neural Networks, etc). However, these visualization techniques do not explain the internal operation/reasoning of a model [116]. Consequently, visual representations of ML models in current VA systems can be made more intuitive than several of the existing visualization solutions such as decision surface view, ROC curves, cost curves, etc. [64]. Furthermore, along with the visual representation, user interaction is very crucial in integrating ML in VA. While the former allows users to visually explore and perceive the data or the model, the later helps the user to demonstrate intent to adjust model performance/behavior by direct manipulation of visual data representations as seen in these works [31, 69, 70, 71, 136].

Conventionally building a machine learning model is a complex process requiring specific knowledge about the theory and application of machine learning processes. A typical workflow of this complex ML model construction includes - data collection, data cleaning, data annotation/labeling, feature extraction, feature selection, feature transformation/preprocessing, and selecting appropriate learning algorithms with associated hyperparameters. Finally, the process ends with hyperparameter tuning to get best performance in terms of accuracy or other defined measures. Evidently, the complexity of model construction is beyond the purview of end-users such as, domain experts or users who do not have the appropriate skillsets or training. Without VA systems, typically any end-user involvement in the model building pipeline is mediated by ML experts. In this process, the mere contribution of end-users can be summarized as: (1) data preparation, (2) answer data-related questions, and (3) review model output mediated by an ML practitioner/developer [10]. This can be changed if visual analytic systems allow power users perform similar operations that are usually performed today by expert ML users through coding or programming techniques.

Holzinger et al. showed four types of ML methods in practice (see Figure 7): (1) Unsupervised ML: methods which does not require human input and is completely automatic, as it does not need a human to label the data. Human experts only review the results of this method in the end. (2) Supervised ML: humans label training data and often select features

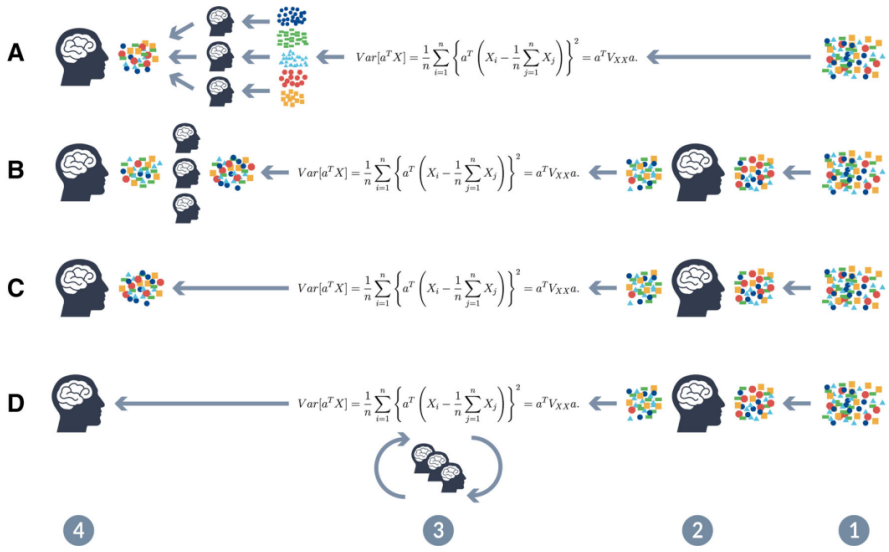


Figure 7: Four different ML-pipelines shown. The colored dots represent input training data, the human head icon figuratively represents the involvement of a human in the model building pipeline. Towards the left is the model output, towards the right is the input to the models. In between lies all the complex computations that drive the models. A unsupervised, B supervised, e.g., humans provide labels for training data and/or select features, C semi-supervised, D the interactive ML approach where humans also actively participate in the logics behind the models using interactive techniques [99].

for the ML models to use to make predictions on a test data set. A good example of this kind is the classification task. (3) Semi-supervised ML: this approach fuses the previous two approaches by learning on data which has a mix of labeled and unlabeled data items. Model learns the data and predicts labels on unlabeled data points based on similarity. (4) Interactive ML or Human-centered ML: this approach includes a human as an active participant in the model construction process. Here humans communicate preferences using interactions, to directly affect the internal operations of a model, i.e., distance or loss functions or specify the weightings of data items or features, etc. (see Figure 7). This is ongoing research where various approaches and methods are being invented, tested, and deployed to solve a number of domain problems, see Chapter 2 for further details. In my research, I seek to investigate this approach further, by including humans in the loop of model construction by designing novel UI and interactive techniques. in human-centered ML systems to empower end-users collaborate with machines to construct robust ML solutions.

1.5 Challenges

While the idea of interactive machine learning is promising in theory, there are various challenges to the design and deployment of such systems, explained further below:

Expertise: Users who need access to ML may exhibit varying expertise/skillsets in ML. For example, while some users are *experts* in ML (e.g., data scientists, ML practitioners), others are *intermediates* who may have programming experience with limited training in data science or ML. (e.g., Gestalt is a visual analytic tool for software developers to debug

models). The third type of users are people who have no data science/ML experience i.e., non-experts in ML. Patel et al. described three difficulties in visual analytics systems for non-experts based on a study that tested ML-based VA's. These were: (1) difficulty in applying iteration in the model exploration process, (2) difficulty in interpreting ML models, and (3) difficulty in gauging ML model performance [170]. As a solution, they proposed to build a library of generic models, that non-experts can use off the shelf for model exploration. Further, a survey conducted by Yang et al. highlighted that designing novel interactions and interfaces for ML tools for non-experts is vital, or else users are prone to make mistakes (due to lack of technical expertise) [248]. Compared to these set of users, expert ML users may also use VA systems to construct models, debug it's performance, or may rapidly experiment with new hypotheses [188, 232]. As such, diversity in users' expertise poses a challenge to the design of future VA systems that provides affordances to model construction/selection.

User preference modeling: In VA systems, user preferences or intents guide how models can be adjusted to specified goals. Thus it is critical to understand how VA systems automatically infer or capture user preferences based on how users interact with these systems [184]. In the past, user preferences have been modeled successfully in various domains such as recommender systems. For example, Middleton et al. described two approaches to model user preferences, which often is also termed as user modeling: (1) Knowledge-based and (2) Behavior-based. The knowledge-based technique builds a static model of users and dynamically matches users with a closest fitting model. On the other hand, a behavior-based approach tracks the behavior of the user when they interact with the system [155]. Based on the usage, this approach utilizes ML-based techniques to predict items users will be interested in such as web pages or products to buy. Other works in user modeling can be found here for reference [6, 125, 135, 172]. While useful, these methods model user preferences that are not applied to systems where the users' primary task is to construct a model. In interactive systems that support model construction, it is a challenge to correctly identify and estimate what kind of interactive controls users wish to have that interactively adjusts models [116]. Furthermore, it is vital to understand how to translate these interactions into mathematical processes that drive ML algorithms to support the desired task. Through my thesis, I seek to address various interactive techniques to infer user preferences to adjust underlying models in VA.

Metric Selection: VA systems integrated with ML models, have shown to perform better than fully automated online model training workflows in supporting user goals. For instance, in this work [8], a physician's expert knowledge is embedded in the data extraction process using a visual interface showing excellent results. However, users of these systems being novices in ML may pick less effective metrics to evaluate or verify model performance. Conventional "accuracy" prediction metric, values each data instance equally, while real application scenarios might wish for selective weighting of data instances based on either cost of mislabelling of important data instances or as specified by the user. For example, in a spam detection problem, classifying relevant emails as spam is more expensive than vice versa. It shows that the cost of misclassification is uneven by data instances. Thus choosing correct metrics to select models is vital to build models that are more personalised and domain-specific. However, it is a challenge to define these metrics that can be used for model selection. Conventionally when ML experts train models they rely on traditional metrics for model selection such as log scores, prediction accuracies, F-measures, ROC

curves, etc. [64, 220, 234]. However, when non-experts or domain experts interact with ML systems, many users utilize perceived percentage accuracy as the only measure to gauge model performance, often leading to selection of sub-optimal models [116]. Furthermore, unlike ML practitioners, non-experts are driven by subjective preferences and preconceived notions (also called domain knowledge). This poses a problem in defining appropriate metrics for model selection, making it difficult to infer what is the right model for the task [116]. Through this research I seek to understand the right metric for power users to evaluate and select a model for various tasks by designing and testing prototype VA systems.

Testing VA Systems: Even though there are established metrics in ML (e.g., accuracy, precision, etc.), there are none which captures subjective choices of end-users. For example, in a design study with biologists for a clustering task [51], we observed that users were driven by their subjective expectations to select clustering models as opposed to rely on known metrics such as, silhouette score, homogeneity index, etc. Lack of appropriate metrics makes it hard to design experiments to gauge the correctness of results in VA systems. One solution to this may be to conduct longitudinal studies with domain experts. However, establishing a relationship with domain experts to evaluate such systems is time-intensive and depends on the availability of such users. In addition, the evaluation results are difficult to replicate and compare with previous studies, due to the subjective nature of the feedback. While processes that includes ML modeling in the context of VA, incorporates domain knowledge to improve model predictions [16] there is no substantial evidence on the effectiveness, or better performance of such approaches over automated/conventional ML pipelines [99]. Through this research, I have collaborated with domain experts [51, 52] to design VA systems that incorporate ML-based data analytic processes to help them accomplish desired tasks. I used the feedback from these design studies as a means to evaluate and further improve the designed VA systems.

1.6 Research Questions

My primary research question is - *How can people communicate their preferences to construct machine learning models suited to their data analytic goals?* To answer this research question I have addressed the following more specific research questions:

- Q1.What are the various techniques of interactive model construction and selection in human-centered machine learning?
- Q2.How effective are multi-model steering and interactive model selection in supporting domain experts to construct clustering models?
- Q3.What are the interactive techniques that empower people translate their preferences into objective functions?
- Q4.How can interactive visual interfaces help users to detect and resolve conflicts in objective functions?

Answering these questions I built new processes to include humans in the ML modeling pipeline by developing workflows in which user interaction data can inform ML algorithms. Specifically, my work guides users in model steering, objective function creation, and model

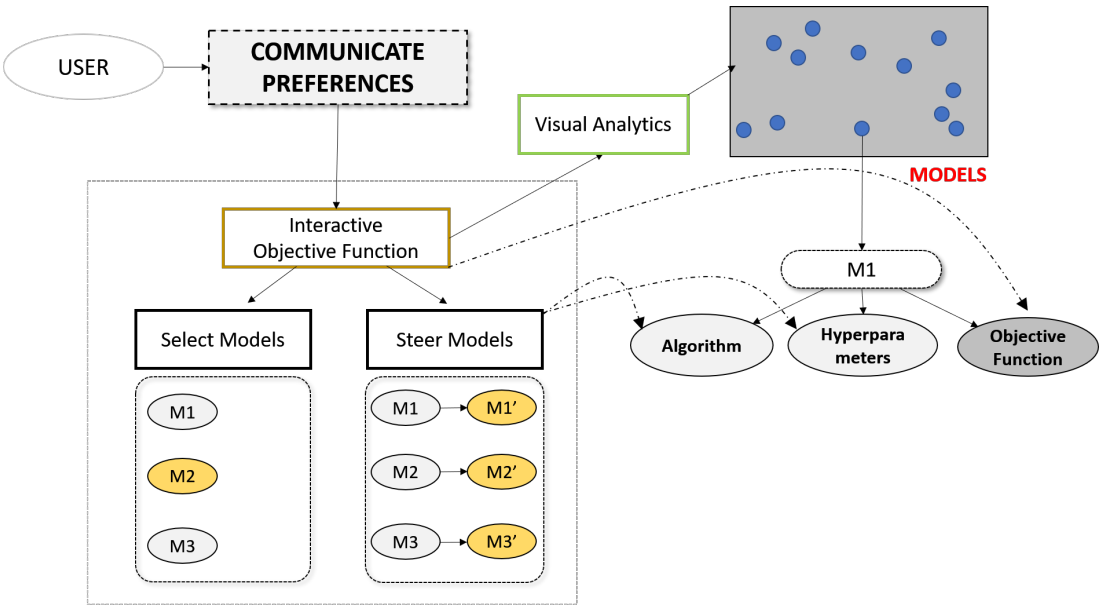


Figure 8: Workflow to empower power users communicate their preferences (data analysis goals) to models.

selection through the use of visual interfaces, without writing code to build supervised (classification and regression) and unsupervised models (clustering) using tabular data (see Figure 8).

1.7 Thesis Statement

New visual analytic systems should allow users to interactively construct, adjust, and select multiple machine learning models with more granular controls (in comparison to current systems such as Auto-ML platforms) to attain personal analytical goals.

1.8 Thesis Outline

Below I outline steps I conducted as part of this thesis:

Studied current interactive ML systems: I started this research by studying visualization techniques, user interactions, and usability issues in current ML-based VA systems. I conducted an extensive literature review to understand the current space of VA systems that allow interactive construction of ML models supporting various analytical tasks, use cases, data types, problem categories, and catering to users with diverse expertise. For example, in a system like Interaxis [121], users drag and drop data points to different regions of a scatterplot, triggering the system to learn a function, which closely imitates the user's goals and constraints. Similarly, another VA system, Clustervision [131] allows data scientists or domain experts with some data science expertise to explore multiple clustering algorithms and steer them based on specified interactions. In VA systems like these, users interact with the visual encodings or data marks to communicate their preferences to adjust the underlying models. Based on the demonstrated interactions the interface updates the visual encodings to reflect the change in the model outputs. Next, users inspect the visual encodings to verify if the model improved or not or if the model confirms to their expectations. This process continues until the user is satisfied by the model. In Chapter 2, I

summarize my research findings from the extensive study of current VA techniques/systems.

Designed and tested VA systems: To seek solutions to various challenges in ML-based VA systems (as described in section 1.5), I designed and prototyped a series of VA systems that enable interactive model construction and selection. While in theory, this may seem promising and relatively straight forward, in practice, while designing these systems, I sought to answer or balance a wide range of questions pertaining to the interaction design, usability, usefulness, and computational feasibility. For example, how do people inspect or interpret output from multiple ML models? How do they select models from a wide range of model options/types? What are the metrics that are meaningful to people that can be used to select these models? What are the metrics to capture the subjective aspects of the user preferences? How do people steer multiple models if a single pre-defined model is inadequate to support their data analytic tasks? These are merely a small subset of questions that needed further investigation, as I brainstormed VA solutions for people needing ML. Furthermore, I evaluated these prototype VA systems by conducting research experiments to find answers to some of these questions. Based on the experiment results (e.g., from quantitative data analysis or qualitative user responses/feedback), I further refined and optimized these systems empowering power users to interactively construct models without the need to program or code or learn extensive data science skills. In this context, though these users may or may not be data scientists, I expected that they should have elementary data analysis skills using tools such as Tableau, Power BI, MS Excel, etc. This ensures that they can relatively quickly learn the interactions in the deployed systems and interpret visualisations that are designed to communicate model outputs and other associated information to make an informed decision in selecting models.

Worked with power users: As aforementioned, the target users for my research are people or analysts who need ML-based data analysis approaches. Furthermore, they may or may not be data scientists, but they have real data, real problems that need ML solutions, and context-specific domain knowledge to inform the underlying ML processes. Seeking to understand how power users interact with VA systems, I followed a design study protocol by developing a VA system incorporating some of the techniques I invented/investigated as part of this thesis (see Figure 2). Specifically, I worked with biologists at the Georgia Institute of Technology who intended to explore a genome (GWAS) dataset by using the designed interactive visual cluster analysis tool called Geono-Cluster. As part of this collaboration, I conducted observational studies and interviews to gather user requirements to further understand the context/problem better. Based on user feedback, I iteratively refined the prototype with the goal to help biologists construct an "optimal" clustering model. I describe this collaborative experience and outcome in detail in Chapter 4.

1.9 Contributions

In this research I have examined two principal interaction techniques, multi-model steering and interactive objective functions. While multi-model steering describes capturing implicit user interactions such as providing labels, weights to data items or features, etc. to interactively adjust multiple models' behavior, interactive objective functions explicitly let users create loss functions that Auto-ML systems can solve for, to find user-preferred models that solve their data analytic goals. In the following I list the contributions of this thesis:

1. This research contributes a series of VA systems and studies on interaction design and algorithmic techniques to understand how end users can interact with multiple ML models using visual interfaces. These studies further confirm interactive processes that allows people to intelligently navigate an overtly large model space to find ML models supporting their data analytic goals.

2. Through this research I validated multi-model steering and interactive model selection techniques with domain experts using real-world data on a year long design study. This study showed how these methods can be deployed in real-world where ML can provide practical analytical solutions.

3. Furthermore, to my knowledge this research is the first to present a visual interface in creating interactive objective functions for multi-model based classifier construction. Extending this work, this research also contributes a novel algorithm and interactive workflow to detect and resolve conflicts in user-defined objective functions.

The above contributions help power users interactively construct and select ML models using VA systems. More importantly, interacting with the visual interface elements, these users can interactively adjust models' behavior until they are satisfied with the models' output that supports their analytical tasks (see Figure 5). Through quantitative measurements and qualitative observations from multiple controlled-lab user studies, I found that the interaction techniques from this research helped people to find an optimal/suitable model from an overtly large space of possible ML models. The problem of finding a suitable model from this high dimensional, infinitely large model space without any computational guidance or statistical method is similar to finding a needle in a haystack. On one end, analysts can navigate this model space by randomly sampling new models and testing their performance in terms of accuracy (or other user-defined metrics). However, random navigation of the model space (an approach similar to finding a needle in a haystack) does not guarantee to find the optimal model for the desired task. To combat that, conventionally ML practitioners/developers navigate the model space using data science principles in search of regions (sub-space of the model space) that contain better performing models [77, 166]. However, through the work of this research, people can interactively navigate this high dimensional model space to find a suitable model for their analytical task.

1.10 Scope and Limitations

My research investigates novel interaction and interface solutions for web-based desktop applications. Any other touch-based platforms, such as mobile applications or speech-based interfaces, are outside the purview of my research work. Further, to prototype and test VA techniques or systems, I scope my research exploration to medium size datasets (e.g., a dataset with few hundred thousand data samples). VA systems that are designed for big data are out of the scope of this work. Furthermore, my research focuses on: (1) Communicating user preferences to ML models, and (2) Interactively constructing ML models for various data analytic tasks. It barely touches on model interpretability towards the end of my research. However, designing systems that supports in-depth model explanations is beyond the scope of this research.

CHAPTER II

RELATED WORK

In this chapter, I discuss various efforts in visual analytics to bring machine learning to the masses. These VA systems range from single to multi to automated systems. Many of these efforts are done in conjunction with domain experts. In the following I summarize literature review of applied machine learning using visual analytic systems.

2.1 Interactive model construction in Visual Analytics

Interactive model construction has been a flourishing avenue of research in the recent past. In general, the design of such systems make use of both explicit user interactions such as specifying parameters via graphical widgets (e.g., sliders), or implicit feedback including demonstration-based interactions or eye movements to provide guidance on model selection and steering. These types of systems build many kinds of models, including metric learning [29], decision trees [236], and dimensional reduction [69, 121, 133]. For example, Jeong et al. presented iPCA (see Figure 9-(a)) to show how directly manipulating the weights of attributes via control panels helps people adjust principal component analysis [105]. Similarly, Amershi et al. presented an overview of the interactive model building process [10].

Stumpf et al. conducted experiments to understand the interaction between users and machine learning based systems [224]. Their results showed that a collaborative shared intelligence-based framework grounded in user interactions could help both users and systems. Stumpf et al. conducted a think-aloud study to understand the forms of feedback humans might give to machines [223]. They found that these included suggestions for re-weighting of features, proposals for new features and feature combinations, relational features, and wholesale changes to the learning algorithm. They showed that user feedback has the potential to significantly improve machine learning systems, but that learning algorithms need to be extended in several ways to be able to assimilate this feedback [223].

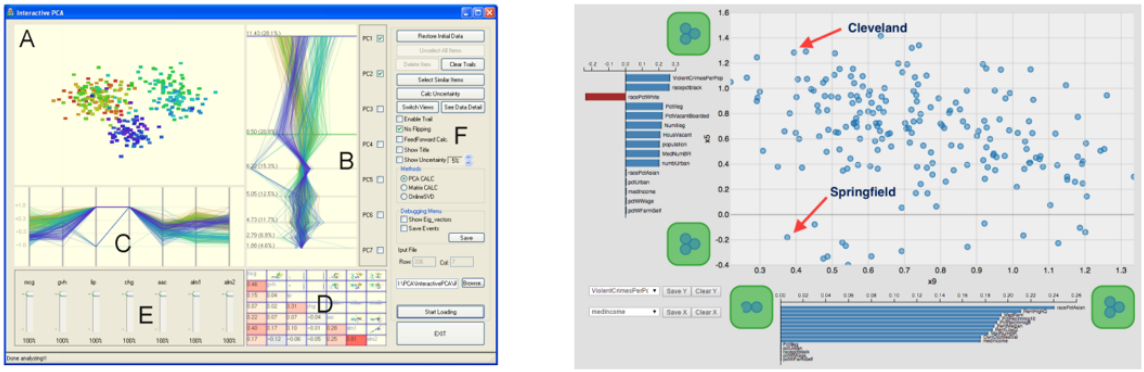


Figure 9: Shows (a) iPCA interactive VA. (b) Interaxis system allowing interaction based scaling of axis on the fly.

One of the core functionalities in these VA systems is the ability to steer ML models interactively. Interactive model steering can be done via demonstration-based interaction. The core principle in these approaches is that users do not adjust the values of model parameters directly, but instead visually demonstrate partial results from which the models learn the parameters [29, 69, 70, 71, 136]. For instance, Brown et al. showed how repositioning points in a scatterplot can be used to demonstrate an appropriate distance function [29]. It saves the user the hassle to manipulating model hyperparameters directly to reach their goal. Similarly, Kim et al. presented InterAxis [121], which showed how users can drag data objects to the high and low locations on both axes of a scatterplot to help them interpret, define, and change axes with respect to a linear dimension reduction technique (see Figure 9-(b)). Using this simple interaction, the user can define constraints which informed the underlying model to understand how the user is clustering the data. Wenskovitch and North used the concept of observation level interaction in their work by having the user define clusters in the visualized dataset [245]. By visually interacting with data points, users are able to construct a projection and a clustering algorithm that incorporated their preferences. There are also work in the literature which shows benefits from directly manipulating a visual glyph to interact with the system, as opposed to control panel style user input [24, 71, 115, 189, 200]. From a user experience perspective, my research aligns closely with these demonstration-based techniques. Interaction techniques prototyped by my research does not presume our users have expertise in model building or steering, but rather let them manipulate the visual results of the models to incrementally refine and steer them.

2.2 Single model based systems

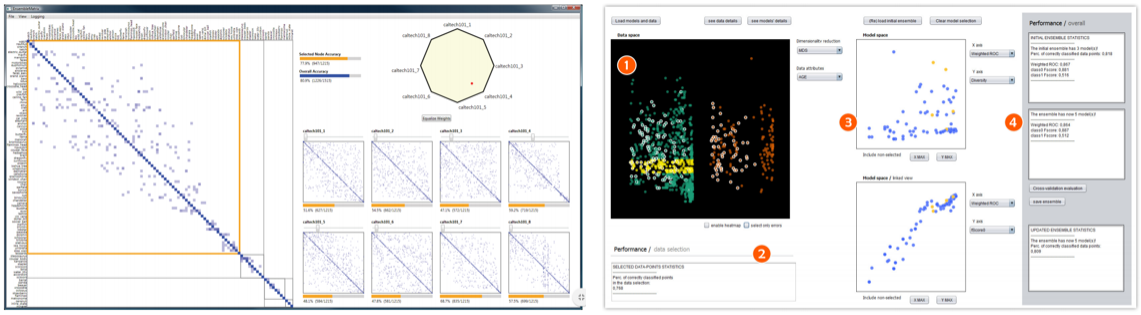
The list of works mentioned above represents single model based VA systems helping non-experts build and adjust model parameters by either a control panel or through UI elements which enables them interactively demonstrate feedback. The spectrum of problem types these systems solve is adequately wide. It includes ranking [240], metric learning[29], decision trees [236], dimensional reduction [69, 121, 133], feature selection [105], weight space exploration [166] and many more. For instance, Podium [240], is driven by a single linear SVM model with the goal to compute attribute weights based on users subjective preference of multi-attribute data items.

In all of these examples, the model infers parameters based on users demonstration of intent by direct manipulation of graphical widgets. Mühlbacher et al. [159] explained increased user involvement in black-box algorithms, using parameter refinement to change the underlying models. Pezzotti et al. [176] have shown a single user steerable model to provide feedback to tSNE models for dimensionality reduction. Similarly, in an interactive recommender system, a user can provide continuous feedback by recording additional choices, or by explicitly scoring (liking/disliking) individual items [104]. My research contribution is distinct from these as I am empowering users to communicate their preferences to multiple ML models (as opposed to a single ML model) with the goal to select a model appropriate for the task and the problem domain.

2.3 Multi-model based systems

Recently multi-model based VA have been explored which comprises of VA's that account for multiple ML models simultaneously to support desired user tasks and goals. Some

multi-model VA systems use multiple models for the same task. For example, Hypertuner [232] looked at tuning multiple machine learning model's hyperparameters. Other multi-model VA systems account for one model for one task, thus supporting multiple tasks in a single VA system. For instance, the system StarSpire from Bradel et al. showed utilizing semantic interactions to steer multiple text analytic models from a set of demonstrations available to users [27]. While effective, their system is scoped to text analytics, handling text corpora at multiple levels of scale. All of these works are very close to my research as they investigate adjusting multiple ML models to support user goals. They also support model space traversal by tuning hyperparameters to find models that best support user-defined constraints.



(a) (b)

Figure 10: Shows (a) Ensemble Matrix allowing users to look at multiple ensemble options. (b) Model Space viewer showing component ML models of an ensemble ML model.

Furthermore, Shneider et al. showed visual integration of data and model space (See Figure 10-(b)), by allowing users identify effective component models on data items from a classification model ensemble [204]. Patel et al. showed an example technique to work with multiple model systems helping users understand the relationship between data, models, and features [169]. Piringer et al. showed an interactive visual analytic system helping multiple regression model comparison and validation in an interactive fashion [179]. Their technique specifically uses a comparison of multiple model outputs to help users select the best model. Similarly, Cutura et al. prototyped an interactive multi-model selection tool focused on the comparison of multiple dimensionality reduction models [190]. Kwon et al. [132] showed a tool to visually identify and select an appropriate cluster model from multiple clustering algorithms and parameter combinations. However, their work targeted data scientists as the user, while we are aiming to build techniques for domain experts without formal data science training.

2.4 Model Space and Model Ensembles

In the literature, there is ample contribution in model space and parameter space analysis to traverse the *model space*. Sedlmair et al. [206] defined visual parameter analysis as a variation of model parameters, generating a diverse range of model outputs for each such combination of parameters. Their work investigated the relationship between the input and the output within the described parameter space.

The topic of model ensembles is related to multi-model steering and model-tuning. Model ensembles increase model performance by fusing multiple model's strength. Different strategies yield different kinds of model ensemble [112, 113]. For example, Potter et al. [182]

showed an interactive ensemble model to allow focus and discovery of simulation outcomes. Datta et al. built a system CommunityDiff, showing a mechanism to visualize ensemble space by using a weighted combination of various algorithms to aid identifying patterns, commonalities, and differences in the space of community detection problem type [56]. Talbot et al. [228] built an interactive ensemble matrix system as seen in Figure 10-(a) visualizing confusion matrices to allow insight gain on various component classifiers. Model ensembles can be built by training the component models on different subset of data [28, 81], or by using different algorithms [130, 246] for each model type (e.g., by using bagging [28]). Through my research I am interested in making contribution in this space by: (1) inventing a technique to search through multiple types of models (i.e., Random Forest models with various hyperparameter and parameter settings), and (2) interpret subjective preferences from user interaction as feedback on all models causing hyperparameter tuning directly changing model behavior in parallel.

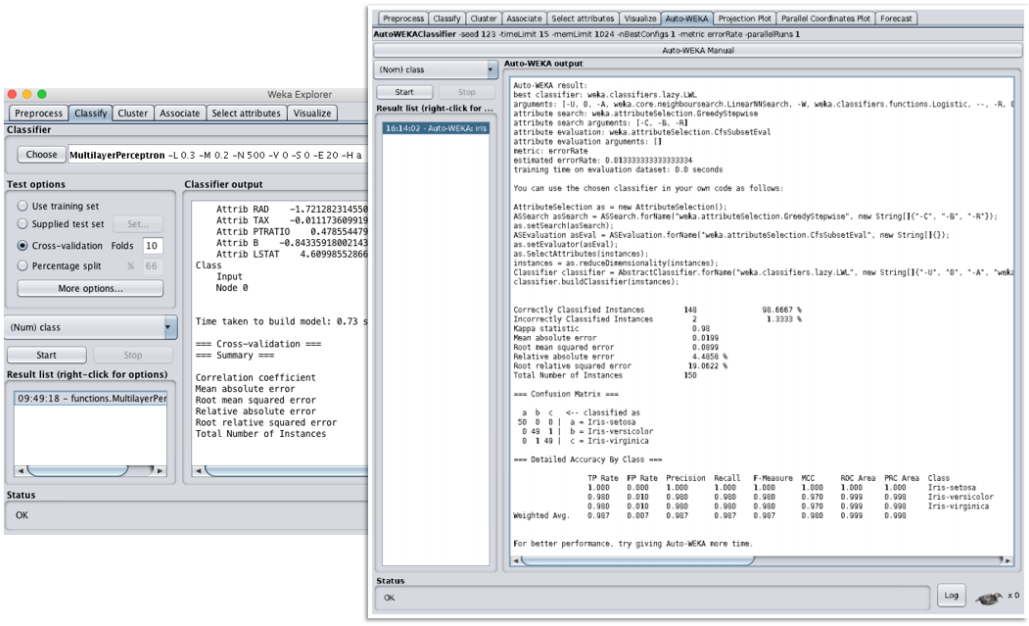


Figure 11: Auto-Weka interface that allows users to automatically build machine learning models by specifying input data and the machine learning task.

2.5 Automated Model Selection Systems

Model building requires selecting a model type, finding a suitable library, and then searching through the hyperparameter spaces for an optimal setting to fit their data. For non-experts, this task can amount to many iterations of trial and error. In order to combat this guessing game, non-experts could use automated model selection tools such as AutoWeka [128, 231], SigOpt [167], HyperOpt [22, 127], and AUTO-SKLEARN [78].These tools execute intelligent searches over the model space and hyperparameter spaces, providing an optimal model for the given problem type and dataset (Refer the interface of Auto-Weka in Figure11). AlphaD3M, is another automatic machine learning system that utilises meta reinforcement learning on sequence models. Their technique uses edit operations performed over ML pipelines to explain the underlying processes [63].

These systems also called Auto-ML is impacting substantially not only to include non-experts in constructing models, but also by helping data scientists be more productive, and efficient. It empowers users to test multiple hypotheses and parallelise their process to search for optimal models. Comparing these Auto-ML systems however, is nearly impossible, as each of these are designed and implemented differently and do not share a common interface or protocol to compare results. Another problem in this space, is to debug a black-boxed Auto-ML system. Usually, developers go through system logs to evaluate the efficiency of the underlying model search process, diversity of models accessed, how well the selected model represents all the class categories in the input data. Analysing logs is a tedious process, and often out of the skillset of users who are novices in machine learning. Large corporations such as Google, build their own internal tools to explain some of these systems. For example, Google Vizier, an internal platform at Google, performs black-box optimization to tune parameters across all various ML models, and further supports explainability to the Google Cloud Platform [88]. However, these tools are all based on optimization of an objective function which takes into account only features or attributes that are quantifiable. This means that such systems ignore the domain expertise of users, which is often not directly encoded in the data. Instead, my work explores how to incorporate this domain expertise through user feedback that is integrated into the model selection process.

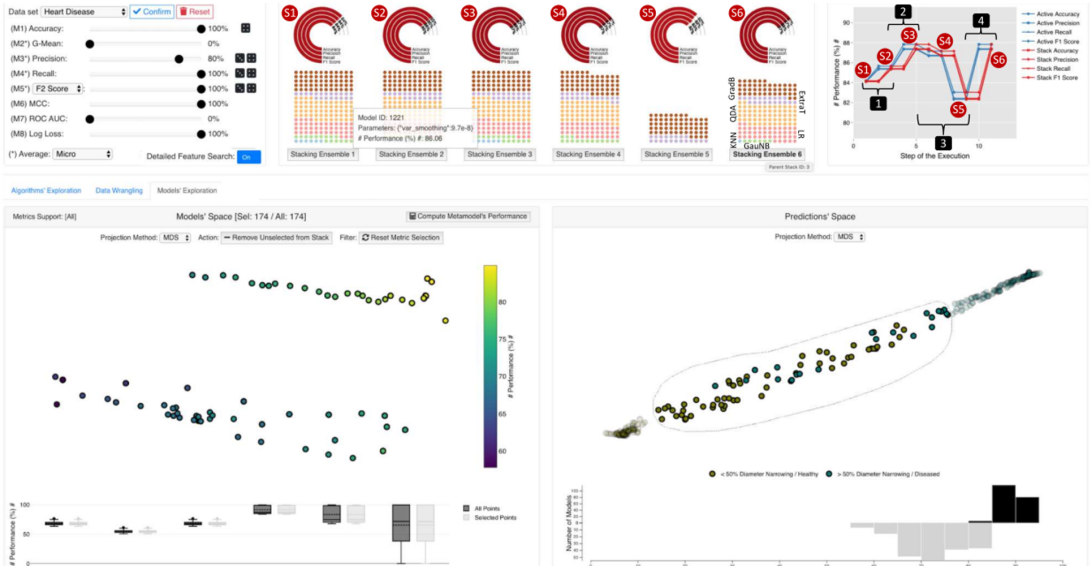


Figure 12: Stacgenvis system empowers users in dynamically managing data instances, selecting the most relevant features for a given data set, and finally interactively selecting models for their problem [39].

2.6 Human-centered machine learning

Human-Centered Machine Learning studies frameworks of machine learning that include a human in the process [11, 12, 198]. A related area of study is the modification of algorithms to account for human intent. Sacha et al. showed how visual analytic based processes could allow interaction between automated algorithms and visualizations for effective data analysis [198]. They examined the criteria for model evaluation on an interactive supervised learning system. They found users evaluate models by conventional metrics, such as accuracy and

cost, as well as new criteria such as unexpectedness. Sun et al. developed Label-and-Learn, allowing users to label data facilitated by interactive visualizations [226]. Their goal was to allow users to determine a classifier's success and to analyze the performance benefits of adding expert labels [226]. Bernard et al. emphasized the knowledge generation process of users performing a visual interactive labeling task, as opposed to conventional machine learning methods [23]. Ren et al. explained debugging multiple classifiers using an interactive tool called Squares [188]. Stackgenvis is a visual interface that empowers users to select optimal models, organize data instances, and select relevant features to train models [39] (see Figure 12).

Holzinger et al. discussed how automatic machine learning methods are useful and discussed their uses in various domains [99]. They noted that these systems generally benefit from large static training sets, which ignore frequent use cases where extensive data generation would be prohibitively expensive or unfeasible. In the cases of smaller datasets or rare events, automatic machine learning suffers from insufficient training samples. They claim such an NP-hard problem can be successfully solved by interactive machine learning via input and assistance from a human agent [99]. This concept of computational models fostering human and machine collaboration is further explored in [49]. Through research investigations, I seek to extend these formalizations by considering human interaction as an estimation of a loss function of the models viewed by the user. In doing so, we generalize human-centered machine learning to multiple models.

2.7 Domain applications of machine learning

Machine learning has impacted many domains, where doomain experts extensively use ML models and pipelines to make informed decisions, in their analytical tasks, such as clustering data to understand associations and relationships between instances and features, or sentiment classification of text data to make sense of peoples' opinion on a topic on social media. In the following, I describe relevant applications of ML in bio informatics and public policy, as I learned through research collaborations with these domain experts.

Bio-informatics: Through my research, I have worked closely with biology researchers to help them interactively cluster data [51]. I learned there are many interactive tools that assists users to interactively cluster data (e.g., [19, 37, 58, 66, 92, 100, 131, 146, 162, 197, 208, 245]), a summary of which is presented here. An early tool to cluster gene datasets is realized in the Hierarchical Clustering Explorer [211] which uses interactive coordinated displays including dendrograms and 2D scatter-grams to support exploration of hierarchical clustering of gene expression datasets. Further, consensus clustering method was shown by Monti et al. [158] as a means to help users in analysis and guidance to select a model from available clustering methods. Their method allows a consensus across multiple iterations of clustering algorithms, in order to evaluate the stability of found clusters. XcluSim [150] is a tool for bio-informatics data helping users to compare multiple clustering results, supporting a diverse set of algorithms. XcluSim combines several small sub-views to form a multi-view layout for cluster evaluation. Another platform, called StratomeX [139] is an interactive visualization application that enables users to explore the relationships of sub-types across multiple genomic data types. StratomeX is mainly designed to support tasks with "comparative nature" (e.g., evaluate how well two or more stratification's support each other). CComViz [107] is a different application that uses the parallel sets technique to compare clustering results. Kern et al. proposed novel methods for evaluating and comparing cluster results and implemented their methods into StratomeX.

Clusterophile [58] and Clusterophile 2 [37] are both designed to enable users to explore different choices of clustering parameters and reason about clustering instances in relation to data dimensions. iVisClustering [134] is a tool that enables document clustering based on a widely used topic modeling method called latent Dirichlet allocation (LDA). Hu et al. [100] and Guo [92] developed interactive tools that enable users to select features while clustering their data. ClusterSculptor [162] is another tool that aids data scientists in the derivation of classification hierarchies in cluster analysis. VisBricks [138] provides multiform visualization for the data represented by clusters (it enables users to select which visualization technique to use for which cluster). While these tools support biologists cluster data, we realized that interactive visual analytic systems needs to be designed to reduce users' cognitive cost as they interact, and enhance interaction expressivity by implementing novel interactions that trigger underlying models to incorporate user preferences, without users' having to go through navigational menus, control panels, or selecting model hyperparameters.

Public Policy/Urban Planning: Recently, I have worked with urban planners who use large scale social media data to mine peoples' opinion and use this information to design new urban policies [52]. Here I summarize, application of machine learning in public policy. Urban planners use large scale social media data [145, 148, 194] to get access to citizens' opinion [26, 140, 161] on topics related to their domain. In this process, they use various ML modeling techniques (e.g., topic models, sentiment classification, text summarization, etc.) and visualizations to make sense of the data and the results from the model [123]. For example, Zhang et al. discussed engaging citizens and other stakeholders in discussion related to spatial planning. In doing so, they demonstrated the application of a web-based toolkit applying hierarchical topic modeling. Their work highlighted three key methods: harvesting geo-social media data from an online resource, identifying text-based social media messages that relate to spatial planning topics, and semi-automatically summarising the contents to explore the themes that appear in the public input.[254]. Other approaches of topic modeling in urban planning using social media data can be seen here [95, 153]. Furthermore, the sentiment classification task has proven to be pivotal for urban planners to understand peoples' sentiment [123, 145, 180, 194]. For example, Paul et al. prototyped Compass, a deep learning based technique of spatio-temporal sentiment analysis from large-scale social media data, on the topic of US Election in 2016 [171]. While these works prove the application and use of ML in the domain of public policy and urban planning through research we realized further work is needed to ensure domain experts access to ML-based technologies which entails being able to adjust models, reason about the models that they select, and interactively navigate the space of model options.

2.8 Interpretability and explainability in machine learning

An aspect of this research helps users to reason about models. While simpler models such as decision trees, or linear regression models are easier to explain or interpret, more complex models such as ensemble models (boosted trees or random forests), and deep neural networks are considerably difficult to interpret. The underlying decision making processes of such models are black-box to the user, which is an open research area in human-centered machine learning. In the following I provide a brief overview of model interpretability to get a better sense of what others in the research community have addressed over the years. A model can be explained by: (1) either using inherently interpretable models (surrogate decision trees, linear models, additive models etc., or (2) using post-hoc analysis methods to analyze

trained deep neural networks or other similarly complex models to help users interpret them [65]. Furthermore, past efforts in post-hoc model interpretation can be categorised as local and global explanation techniques. Local explanation techniques show a model's reasoning process in relation to each data instance. Global explanation techniques aim to provide an understanding of the models' behaviour as a whole and analyze what knowledge has been acquired after training.

Interpretable models: This category includes models that are inherently interpretable, such as decision trees, rule-based models [137], additive models [34], sparse linear models, etc. [235]. Compared to neural networks or ensemble models, these models comprises of internal components that can be directly inspected and interpreted by the user. For example, users can probe various branches in a decision tree, or visualize feature weights in a linear model. While these models help users make sense of the model predictions, the performance accuracy falls behind compared to state of the art complex models such as deep neural networks. However, owing to the benefits of simpler models, recently a number of neural network architectures also utilises interpretable components such as attention modules [247] or prototype layers [38, 96, 143, 156] for easy human interpretability. As useful these additions are to the network architecture, these models may need to balance between model performance (i.e., accuracy) and interpretability.

Locally explainable models: This approach includes explaining a pre-trained ML models' reasoning process with respect to input data instances. In this space, a frequently used technique is to calculate and visualise feature attributions [13, 17, 73, 149, 185, 209, 216, 217, 227]. Feature attributions can be computed by slightly perturbing the input features for each instance to verify how the models' prediction response varies accordingly [126, 217]. In the context of deep neural networks, feature attribution can be computed by back-propagating through the network [209]. Another technique in this category includes sampling features in the neighborhood of an instance to compose an additional training set. An interpretable surrogate model is re-trained using the same training set such that it mimics the original models' performance accuracy. Using this approach an original models' prediction can be explained by an interpretable model (e.g., linear regression) that is relatively easier to inspect [192]. A major flaw in this approach is that local explanations are shown to be less reliable and consistent as the explanations holds true only for a specific set of data instances. The explanations do not hold true for other similar data items in the training set. In other words, for two data instances from the same class label, the explanations may strikingly contrast from one another. In addition, it could also be badly impacted by adversarial perturbations [84, 122] and confirmation biases [5]. Another drawback of this approach is that users have to go through the tedious process of manually inspecting each data instance to make sense of the model.

Globally explainable models: This approach focuses on explaining models' behavior by showing a global overview, rather than describing predictions of local instances or input regions [65]. For deep neural networks, a particular set of global model explanation techniques focus on understanding the latent representations learned by the neural network through activation maximization techniques [252] which calculate inputs that can maximally activate each individual neurons in intermediate layers in a neural network. There are also concept-based explanations that show how models' makes predictions globally by recovering relevant concepts [85, 120, 249, 256] that are understandable to humans. For example,

Figure 13: Concepts learned from input images data using the concept activation vector technique [120].

the technique interpretable basis decomposition (IBD) explains image classification model by showing relevant concepts that are human-interpretable [256]. In particular, concept activation vectors (CAV) are discussed by Kim et al. [120] (see Figure 13) as a framework to interpret latent representations in deep neural networks. This technique has been shown to be implemented by using supervised approaches where data with human-annotated concepts is available [120], or by unsupervised techniques (e.g., clustering) to retrieve relevant concepts directly from the training data [85].

2.9 User Preference in Objective Functions

In machine learning, users solve various problems which are context-dependent and personal [7]. For example, a problem scenario in ML to enable personalized interactions of a robot with autistic children [196] is entirely different from that of constructing a classifier personalized for patients with an Alzheimer's disease [74]. Diverse problem scenarios create an opportunity to specify a diverse set of user preferences. These preferences are the building blocks to construct an objective function. We studied the literature to understand what kind of specifications users can provide to construct a robust ML model such as a classifier [43, 233, 248, 258]. The following summarizes relevant past efforts to capture novel user preferences. Kapoor et al. discussed, often users have to rely on the overall classification accuracy of predictive models instead of relying on predictions generated by marginal models. Marginal models compute accuracy by taking the number of correctly labeled instances

of test data divided by the total number of instances the model has classified [116]. This often leads to a bad model selection. Zhu et al. described the machine teaching paradigm where a machine teacher (usually a domain expert) shows informative data instances of positive and negative class labels to maximize the distance between the classes[258]. These specifications directly affect how a model learns from the training data.

Lime, a submodular optimization technique, helped users to interpret models by explaining the prediction of a model on a set of data instances [193] that are relevant to them. Tamuz et al. showed an adaptive algorithm that estimated a similarity matrix from human judgments based on comparisons of triples [229]. To paraphrase, the authors asked users - "is object A similar to B or C?" . Applying the same ideology in classification tasks a user can specify data instances that similar and should be predicted in the same class label or vice versa. The system Flock asked crowd workers to define the reason behind a pair of instances to be in a positive class and vice versa. Their method captured features specified by crowd workers when automated feature extraction was not feasible [45].

Kapoor et al. discussed if users can understand the model behavior, they can assess the possible next moves to adjust the model further [116]. For example, users can evaluate a model if it correctly predicts similar data instances in the same class label or not. If certain data instances are not in the same class label, users may provide additional examples in order to refine the model's characterization of the data. We realize that unlike conventional model building pipeline which relies on metrics such as precision, recall or cross-validation scores, non-expert users can assess the quality of models based on preferences they specify as part of an objective function.

2.10 Many-objective optimization systems

Various techniques have been used to visualize solution sets from an objection function space such as MDS, RadViz, Bubble chart, Parallel coordinates, Self-organizing map, etc. A detailed comparison is provided here [98]. Further, He et al. proposed a new visualization technique to map solutions from a high-dimensional objective space to a 2D polar coordinate plot. Their method helped understand trade-offs between objectives and find desirable solutions [97]. Sahu et al. showed the use of a radar chart to visualize many-objective solution spaces [199]. Walker et al. visualized a set of mutually non-dominating solutions They used Radviz visualization to show multi-objective solutions and introduced techniques to measure the similarity of non-dominating solutions [239].

Many researchers have looked at measures to assess the diversity of Pareto-optimal solutions in multi-objective optimization problems [141, 142]. While these works have looked at visualizing the solution space (mostly non-dominating Pareto-optimal sets) in an objective function, we are interested in visualizing only a subset of solutions of the order of $k = 1to15$. We thus used two effective visualization techniques to display the Pareto-optimal ML models - (1) Parallel coordinate plot and (2) Radial chart or Star plot view. Further, we intend to help users construct the objective function as opposed to only view its solution space. To our knowledge, the construction of objective functions by non-experts using interactive visualization techniques is missing in the literature.

2.11 Conflicts in multi-objective objective functions

Below we summarize, a set of works from various domains where people have addressed conflicts in objective specification by incorporating various learning techniques. In many

practical real world ML applications such as in finance, transportation, engineering, medical diagnosis, etc. requires addressing multiple user goals that may often conflict with one another [144]. Often these goals are specified to the system using a multi-objective objective function representation [47, 175]. In such multi-objective objective functions, it is considered as 'k' (number of objectives) increases, the power of finding dominant solutions diminishes as satisfying each of the objectives becomes mathematically intractable [57]. Purshouse et al. confirmed that multiple objectives may be conflicted with one another; resolving conflict may show better performance in one objective than others. However, in some cases they may also show harmony that both objectives sees improved performance [183]. A few approaches to address better performance with a large number of conflicted objectives includes: (1) multi-start strategies of the optimization process [101], and (2) purely random search for objective functions with more than 10 objectives [124].

In this context, Zhang et al. defined conflict analysis as a method to find conflicts, reason about it, and then resolve it [253]. Bell et al. further describes conflicts in decision making and provides an overview of quantitative approaches to address them in optimization problems [21]. Reed et al. explored scatterplot charts to visually inspect the set of conflicting objectives to solve a ground water monitoring and optimization problem [187] (e.g., discovered a conflict between cost and uncertainty). In machine learning, there is a recent interest in multi-objective machine learning optimization functions, which tackles conflicting user objectives. For example, minimizing the number of features and the maximizing feature quality are two conflicting objectives. In model selection, there is the conflict between model complexity and model accuracy (more complex, more accurate the model) [108, 109]. Multi-task learning is another avenue where multiple tasks are solved jointly using a multi-objective optimization paradigm; however these tasks often conflict with one another that needs a tradeoff analysis [210]. A common solution is to utilise a proxy objective to minimise a weighted linear combination (per task) of loss. Sener et al. showed a solution to the conflicting objectives by solving for pareto optimal solutions [210]. We observed that there are many areas in decision making, and in machine learning where conflicting objectives play a critical role. In the past, authors worked around it using elementary visualisations as a means to allow users perform tradeoff analysis, were users inspect a set of pareto optimal model options. More importantly, none of the work in the past has specifically worked towards conflict detection and resolution in objective functions, specifically designed for machine learning model selection, a problem that we specifically solve in this thesis.

CHAPTER III

MODEL CONSTRUCTION AND SELECTION IN VISUAL ANALYTICS

Q1: *What are the various techniques of interactive model construction and selection in human-centered machine learning?*

This chapter describes my research to answer what are the various approaches to interactive model construction and selection in VA systems? First, I explain the concept of model selection and then I describe the spectrum of model selection in VA systems. Further, I describe two model selection techniques that I prototyped and evaluated. Finally, I summarize how people select models using interactive visual interfaces.

3.1 Model Selection and Model Steering

3.1.1 Types of Model Selection

The choice of model is critical in an interactive ML workflow. What if the model chosen is sub-optimal for the task, dataset, or question being asked? What if instead of parameterizing and adjusting a chosen model, a different model provides a better fit? Thus a VA system needs to deploy computational techniques to select models that non-experts can use. This process is called *model selection*.

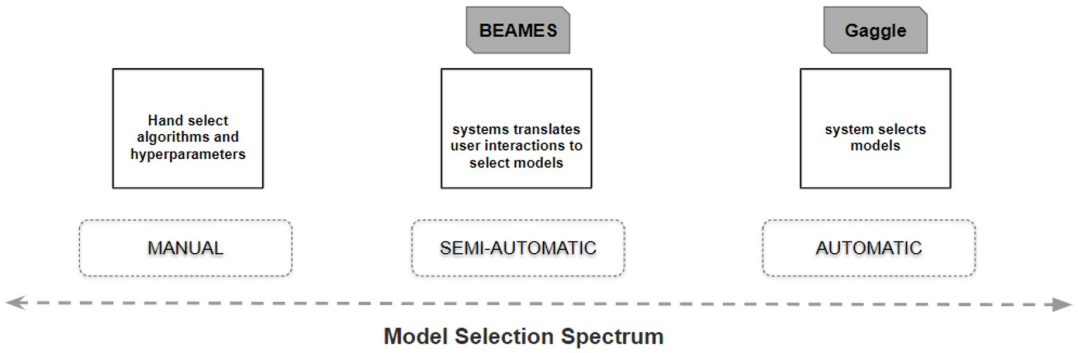


Figure 14: Various ways people select models in visual analytics.

There are many ways by which a model can be selected (refer Figure 14) from the overtly vast model space further explained below:

- **Manual Model Selection:** In this method, the task of model construction and selection of an optimal model is passed on to the users. Users manually construct models by picking a learning algorithm and a set of hyperparameters either from a control panel style user interface or by writing programs/scripts through a text editor. They evaluate different models by reviewing their output on an underlying dataset

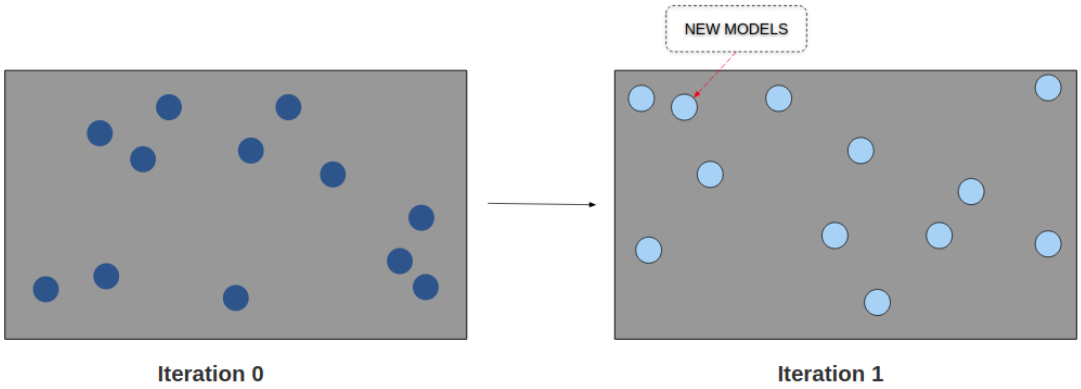


Figure 15: In manual model selection, models are sampled from the model space by manually specifying hyperparameter settings. At the next iteration, totally new models can be sampled by manual specification of new hyperparameter values.

(train and test) or by inspecting models performance on a chosen metric (i.e., cross-validation score, F1-Score, etc.). While useful, this approach is traditionally followed by ML practitioners/developers who know how to code/program and understand data science/ML theories (refer Figure 15).

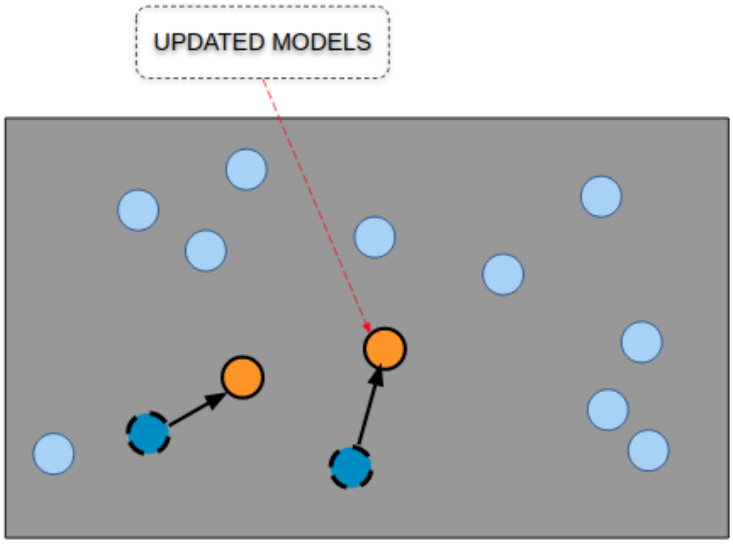


Figure 16: Semi-automatic model selection changes the models in the model space per iteration by configuring new hyperparameter settings using model steering approaches.

- **Semi-Automatic Model Selection:** In a semi-automatic model selection approach, a user interactively constructs models and adjusts their performance until they are satisfied. In this approach, a model solver in a VA system performs model construction, hyperparameter tuning, and model selection based on a pre-defined metric. Often the model metrics are inferred based on demonstrated user interactions. In some systems, users can interactively specify model metrics to guide the system to select models aligning with their preferences and choices (refer Figure 16). In my research, I deployed a semi-automatic model selection technique in a VA tool called BEAMES

[50], aiding selection, inspection, and steering of multiple regression models from the model space (explained later in the chapter).

- **Automated Model Selection:** Automated model selection is facilitated by auto-ML platforms such as, [22, 127, 167]. For example, Auto-Weka [128, 231] provides an auto-ML classifier, which requires users to input a dataset, and provide the attribute name of the target label. Next, the auto classifier searches within a domain(an input) range of each hyperparameter values to find a model with the right combination of hyperparameters such that it maximizes or minimizes the specified metric to evaluate the model such as, cross-validation score, training accuracy or testing accuracy, etc. Next, Auto-Weka iterates until the maximum number of iteration limit is hit, or the chosen metric does not change per iteration. Finally, Auto-Weka responds with an optimal model and the predicted label output for training and test set. There are other auto-ML tools which non-experts can use such as Hyperopt, Sigopt, BigML, etc. (refer chapter 2), all of which follow a similar workflow.

3.1.2 What is Model Steering?

In a semi-automatic model selection approach, users interact with VA systems to adjust model behavior; this phenomena is explained by *model steering*. Precisely model steering helps systems to interpret and translate user interactions into actions that change the underlying numerical processes that drive these models. Various user interactions contain critical information about the users thinking processes, approaches, and the path they traverse to derive insights [61, 178]. Some real-world applications of a model steering approach can be seen here [83, 154, 177, 215]

Along the same lines, Liere et al. have defined computational steering as a process to enable users to change parameters of simulations on the fly [160, 237]. Their work emphasizes the concept that simulations run over many iterations, where users may need to update parameters before completion. We ground our concept of model steering in this prior work and refer to it as a process in which a model's parameters are changed to produce updated results iteratively, and multi-model steering as a process in which a model's hyperparameters, and parameters are changed (see Figure 17). Model parameters refer to values that are learned during training of a model, such as coefficients, while hyperparameters are typically determined through some process external to training, such as cross-validation.

Thus model steering helps people incrementally build machine learning models that are tailored to their domain and task. Existing visual analytic tools allow people to steer a single model that is pre-defined by a system developer or a ML practitioner. However, if this single pre-defined model is inadequate to correctly characterize the data (poor fit to the underlying data), users may see sub-optimal performance/results. In my research, I am investigating novel techniques to search the model space based on inferred user preferences to find models that satisfies users subjective requirements.

I begin by investigating semi-automatic and automatic model selection in VA both of which uses interactive model steering. BEAMES, a VA tool that I protoyped presents a technique to allow users to inspect and steer multiple machine learning models [50]. The technique steers and samples models from a broader set of learning algorithms and model types. The system allows users to perform regression via a multi-model steering and a semi-automatic model selection approach, explained further below.

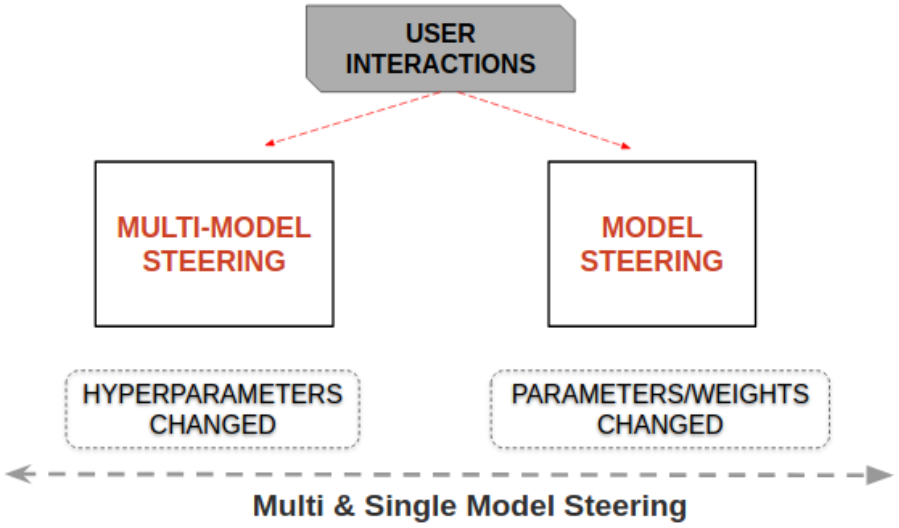


Figure 17: Types of model steering as seen in multi-model and single-model based VA systems.

3.2 BEAMES - inspect, steer, and select regression models

The visual analytic technique presented in BEAMES allows domain experts to inspect regression models by checking a model's predicted output on a tabular data. The underlying technique in BEAMES searches the model space for models that more closely adhere to data items and attributes the user is interested in. This human-in-the-loop process allows domain experts to explore a myriad of models for a regression task, and add domain expertise into the model building process to produce models which adhere to their subjective and objective preferences (see Figure 18).

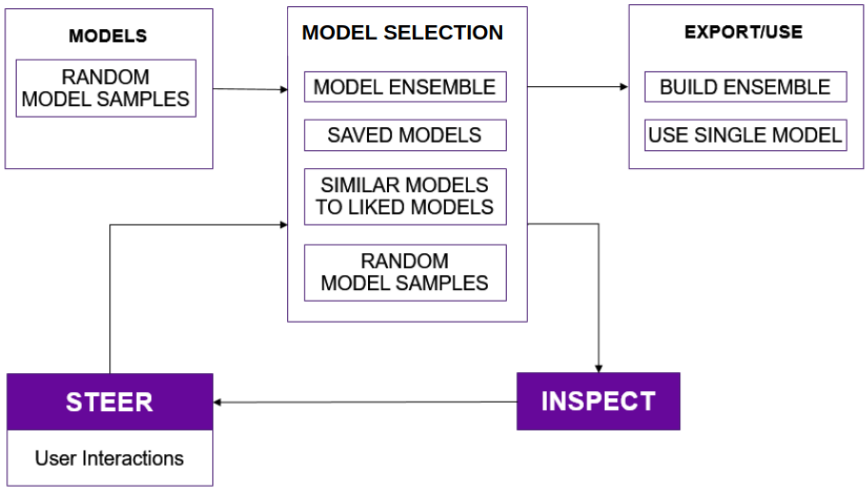


Figure 18: Working process of BEAMES incorporating model steering, inspection and selection for a regression task.

BEAMES is designed to help users define critical data instances on which the performance of a model is crucial. Accurate prediction of critical data instances can increase

user's trust in the model. Our technique allows people to steer and inspect multiple models. BEAMES assists the inspection process by recommending models (from a collection of models) which successfully make predictions on the critical data instances with zero or relatively low error value. Showing a wide spectrum of models for the given regression problem can be beneficial to domain experts who otherwise would not be aware of the many possibilities and permutations of models. Being able to filter the data by instances and filter models by their performance (by simple double range sliders and toggle switches for categorical items), users can drill down to models which are successful and can validate them by checking their results on critical data instances. Further, users can add domain knowledge in this model construction process e.g, they may specify features that are more important than others, or data instances that are more important to correctly predict.

The technique prototyped in BEAMES has the following primary components: i) interactive weighting of critical data instances, ii) interactive feature selection with weights, iii) interactive model selection, and iv) building model ensembles.

3.2.1 User Interface

The user interface of BEAMES consists of four primary views: a data table, a model view, a control panel, and a model detail view.

Data Table: Users can see training, test, and hold out set in the data table view which follows a standard spreadsheet style (See Figure 20-(b)). The columns show three-state toggle switches enabling users to emphasize, de-emphasize, or discard an attribute. Further using a slider they can specify attribute weights. Users can specify certain data instances which they think should be correctly predicted by the model, also called critical data instances. Hovering over any data instance, the system shows which models correctly predicted this instance.

Model View: Each regression model is shown as a circular glyph in the model view (encoded color shows the average residual error, see Figure 20-(a)). Hovering over any of the circle shows model details such as its learning algorithm, mean squared error, number of data instances correctly predicted, etc. Clicking on a circle allows users to inspect the model, as it adds the predicted value column in the data table view (e.g., housing price on a housing data set). Users can interact with these models in numerous ways as explained in the next section. Further inspecting a model opens the model detail view for detailed analysis.

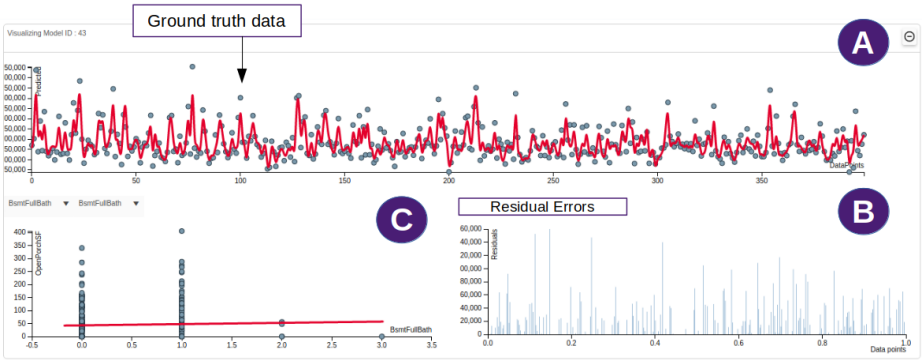


Figure 19: (a) Shows the model detail view. (b) Bar chart showing residual error per data instance. (c) Scatterplot showing relationship between two selected data attributes.

Model Detail View: This view shows a scatterplot with a line showing how well the model fits the data. Further, it shows a bar chart highlighting residuals at data instance level. Finally users can also inspect relationship between a pairs of attributes through a scatterplot (See Figure 19-(a)).
Control Panel: The control panel contains frequently used operations such as filtering data instances, and checking attribute weights. Models can be filtered by model accuracy, average residual scores, or desired number of correctly predicted instances (See Figure 20-(c)).

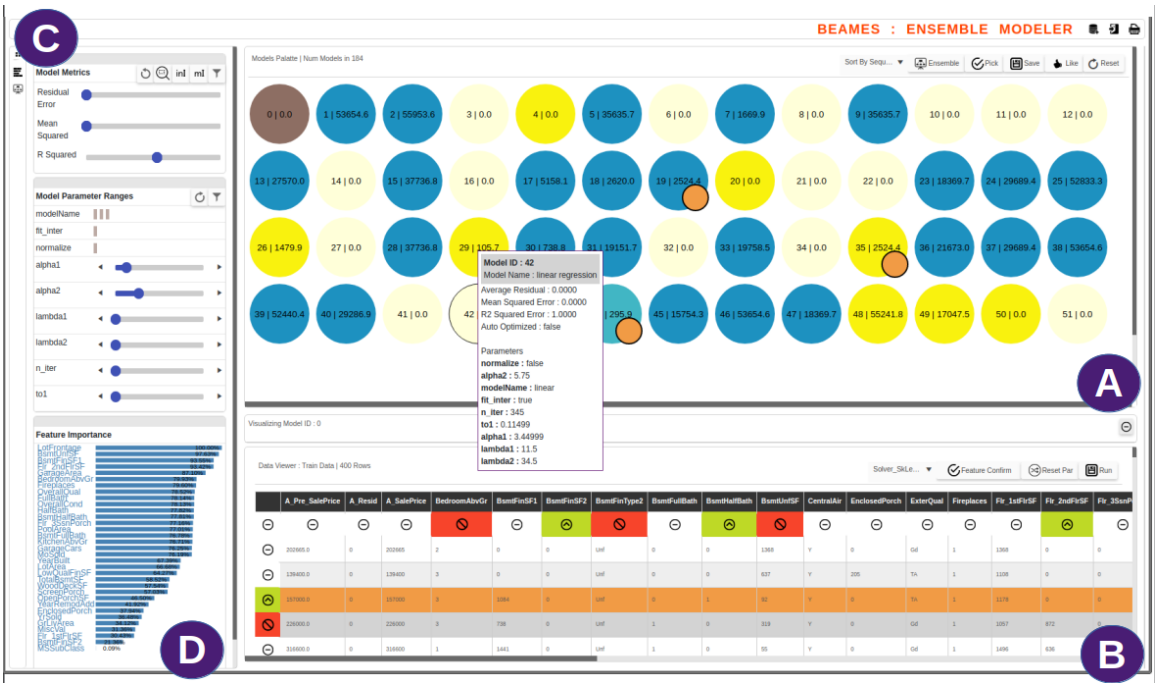


Figure 20: The BEAMES user interface for multi-model steering, selection, and inspection for regression tasks. The model view (A) shows circular glyphs representing regression models color coded by residual error. The data table (B) shows training, test, and application data sets. The control panel (C) allows users to filter models and critical instances, and change feature weights (D).

3.2.2 User Interactions

User interactions in BEAMES are designed to update the underlying models via both interactive model steering and selection. This section describes these interactions.
Save Models: Users can save any model M_i in the model view. The system saves its learning algorithm L_i and the set of hyperparameters represented as $[\lambda_1, \lambda_2, \lambda_3 ... \lambda_m]$. At each iteration, BEAMES shows saved models with an orange stroke aorund it as seen in Figure 21-(b).
Like Models: Users can like a model M_i in the model view (see Figure 21-(d)). In response the probability p_d of the learning algorithm that produced that model is increased by a factor r_f. We randomly set the value of r_f by using a threshold ϵ. With trial and error, we found $\epsilon = 0.1$ showed promising results (certainty to select model's with higher accuracy is more). Likewise, the hyperparameters λ_i of that algorithm are sampled from within a threshold region of the hyperparameters used in the liked model. This ensures that

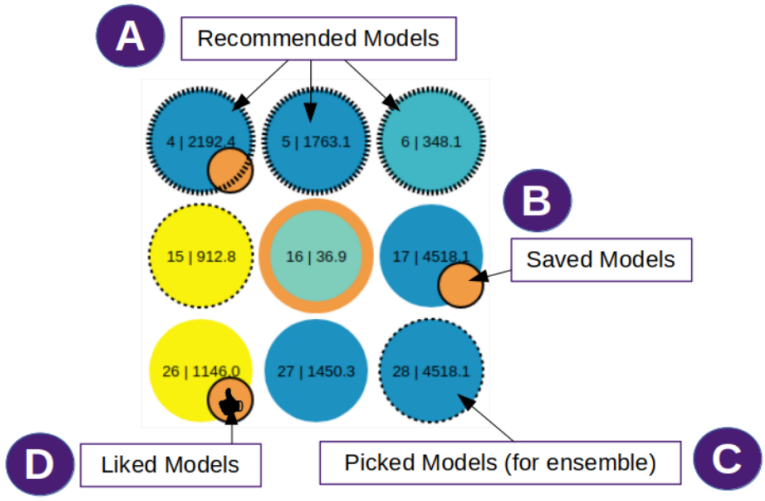


Figure 21: Circles represent regression models. (a) Recommended models to the user (b) Saved models by the user (c) Models picked by the user to create an ensemble model (d) Models liked by the user.

a large share of the new sampled models are from the neighboring regions of the model's users liked. However, the technique still randomly samples other model types to ensure diverse choices in model selection.

Adjust Data Instance or Attribute Weight: Users can adjust the weights of data instances and attributes by adjusting the respective sliders (See Figure 24-(a)). As a result, all the available M models are retrained using N training instances with user specified features A_k, where $A_k \subset A$, W is user-specified feature weights, and Ω is data instance weights.

Ensemble Models: Users can select (by a pick interaction) G models to build a model ensemble (See Figure 21-(c)). The system uses each component model m_j to build a model ensemble E. Using a bagging technique from [28], BEAMES assigns a higher probability to sample data instance d_i, whose weight ω_i have been increased by the user. The final ensemble model E's output is the weighted average of the predictions of the models in the ensemble. Further, when users are satisfied with a model, they can directly export the model for future use.

3.2.3 Usage Scenario

Amy is a real estate agent who reviews existing and new properties to analyze their market prices in the city. However, not being a data scientist she is not conversant with complex modeling techniques, which can help her accurately predict property prices or property ratings in the future. Amy loads the Ames, Iowa housing dataset [102] in BEAMES. The data set is automatically split into training (750 samples, 36 attributes) and test set (200 samples, 36 attributes) by BEAMES. It has a target attribute namely *SalePrice* that contains the property price of each house. Every row in the data is a property (a house) described by attributes such as *property size*, *fireplaces*, *year built*, *number of bedrooms*, etc.

Next, BEAMES builds 64 regression models, each randomly sampled using a combination of learning algorithms (linear, ridge, and bayesian regression) and hyperparameter values (alpha, lambda, tol, etc.). As Amy is not formally trained in the specifics of the

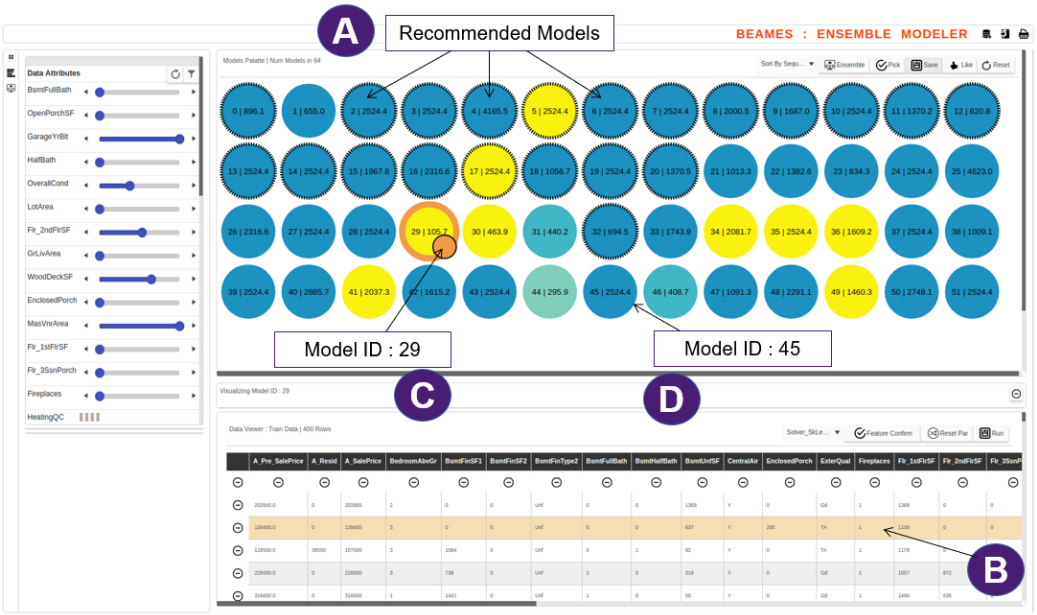


Figure 22: View showing Amy loads BEAMES to find 64 regression models. (a) Recommended Models (b) Amy hovers over a critical data instance. (c) Amy clicks model 29 and saves it. (D) Amy clicks on model 45 and inspects the output.

models, she begins her exploration by how well specific models predict property sale prices. In the model view (See Figure 20-(a)), she sees the collection of models as circular glyphs color encoded by their residual error scores. Browsing the colors of the circular glyphs (representing models), Amy decides to start inspecting the yellow colored ones, which has lower average residual error.

Amy inspects model 45 (Avg. Residual Error of 521, see Figure 22-(d)), as it has accurately predicted over 300 entries on her training data while errored on the rest of the entries from the training set. Next, she inspects model 29 with a much lower average residual error 105.7 and finds that most of the data instances are predicted accurately (See Figure 22-(c)). However, to double-check if some of the known data instances were correctly predicted, she uses the filter panel on the left. She sees that the model 29 did not correctly predict most of these data instances which are critical to her.

By hovering over these critical data instances (See Figure 22-(b)), BEAMES shows models that Amy should inspect, as they made correct predictions on those instances (See Figure 22-(a)). Amy reviews a few of the suggested models. She finds that the recommended models performed better for the critical instances, though they had higher overall average residual errors. Next, Amy clicks on the three state toggle button on these data instances to emphasize them and specify sample weight using a slider. In addition, Amy thinks the property price should be most strongly defined by the $numberofbedrooms$, $garageArea$, $2ndfloorarea$ attributes. She again uses the slider to increase their weight, while reducing the weight on $frontPorchSize$ and $DrivewayQuality$.

BEAMES updates the model view with newly computed models. Amy quickly looks over the collection to find many models have scores close to 0. She finds two models from these which correctly predicted the critical data instances. She saves them from the collection. Next she evaluates them on the test set. Clicking on these models, Amy finds

the prediction on the test data is a bit off. For example, property id. 104 shows a predicted price of $141,345, while the actual price is $99,322.

Confused to find the relatively poor performance on the test set, Amy uses the control panel to see the importance of the features in the horizontal bar chart. She sees the relatively strong weight on the *numberofbedrooms* attribute (as she intended previously). She adds increases weightings on few other relevant attributes i.e, *overallPropertyRating*, *numberOfFloors*, *distanceToTransit*. Next, she discards a few training data samples thinking the prices on those properties are noise in the data. She likes a few models based on their performance. Further, she picks a few models to create an ensemble from these liked models and generates more models.

Amy browses the newly computed models. She sees a brown colored circular glyph, representing an ensemble model built from the models she picked. She clicks on it and finds that it shows a very accurate prediction on the training data. Amy confirms the same on the model detail view, as the line fits the set of points (representing actual ground truth values). Similarly, she sees almost 0 average residual error from the bar chart view. Amy is happy with the models and saves a few (including the model ensemble). She loads a new data set with same attributes (See Figure 23-(a)). She clicks on the saved models to see predictions on the new dataset (See Figure 23-(b)). At this point, Amy has models to help her predict house prices, which informs her how housing prices may change.

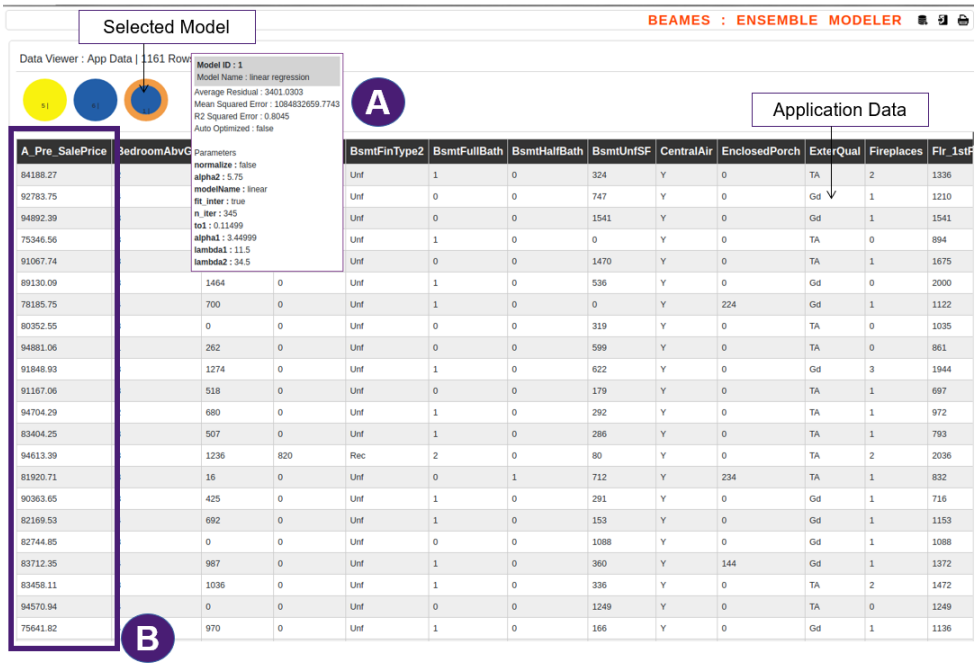


Figure 23: Amy loads the application data set to apply the saved models and see predicted output. (a) Horizontal panel storing models saved by Amy. (b) Predicted output ("sale price") when a model is selected.

3.2.4 Technique

Data: We define our full data set as C data instances, which is then split in N training, K test, and B application data instabces ($C = N + K + B$). Users train models on the training set D containing N data instances, then validate on the test set T containing K

data instances. When they find an acceptable model they export it or use it on application data set H containing B data instances.

Model Sampling: We define a *model* M_i as a function $f : \mathcal{X} \mapsto \mathcal{Y}$, mapping from the input space $\mathcal{X}$ to the prediction space $\mathcal{Y}$. Here, the prediction space is $\mathbb{R}$ and each model m_i is a regression model. For the modeling algorithms we used Scikit Learn's machine learning package [32]. Each model is sampled by combining a learning algorithm l_k from a set of J algorithms (hand picked by us for the regression task). Tested algorithms include Linear Regression, Ridge Regression, and Bayesian Regression. Each learning algorithm comes with their own set of hyperparameters λ_m. A sampled model is defined as $m_i \mapsto Model(l_k, [\lambda_{k1}, \lambda_{k2}, \lambda_{k3}....])$.

The system initiates with randomly sampled S models. We'd like the sampling distribution to be uniform across algorithms such that users can inspect a wide spectrum of model outputs for the given regression problem. For that reason, we initialize probability p_k to sample a learning algorithm l_k (for a model m_i) from J possible models as $1/J$.

Updating training data: Users can load training data D on the data table. Every data sample is initially set to an equal weight of $\omega_i = 0.5$. However, users can interactively set weights on the samples between 0 and 1. 0 meaning to discard the data sample in training, while 1 is to place the highest strength to the learning from the sample.

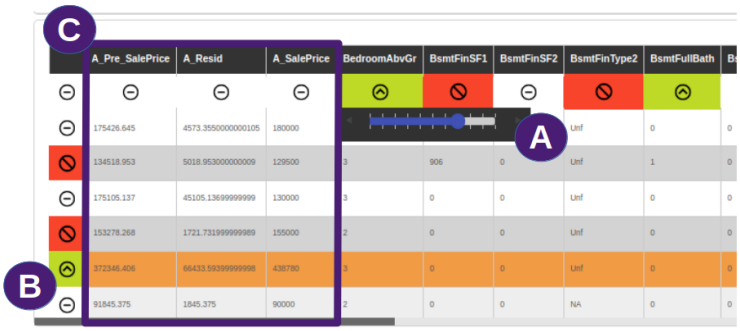


Figure 24: (a) The data table view showing the toggle switches (green, white and red) for features. The sliders allow users to add weights to the data samples and attributes.(b) Similar toggle switches and sliders for data samples. (c) From left : Column 1 is the predicted output, Column 2 is the residual error, Column 3 is the ground truth value to predict, in this case sale price of a house

User driven feature engineering: Similar to the weighting of data instances, users can emphasize, discard or weight quantitative features (see Figure 24-(b)). Using the UI toggle buttons, users can specify if they want to emphasize or discard a feature for model training (see Figure 24-(a)). Discarding a feature removes it from the set **A**. Emphasizing a feature reveals a weight slider, which the user can set between -1 and 1. Setting a weight of -1 enforces the model to place a higher emphasis on lower values of the attribute than others.

The instance and attribute weights assigned by the user directly affect the computation of the y dependent variable similar to the work by Cleveland [46]. The loss for each model is a weighted least squares loss. Thus, the different regression models solve the following regression problem:

$$\min \sum_{i=0}^{N} \omega_i * (\hat{y}_i - y_i)^2$$

where,

$$\hat{y}_i = b_0 + \sum_{i=0}^{M} b_i * x_i * w_i$$

ω_i is the user-defined weights for data instance i, b_0 is the intercept, b_i is the coefficients of the attributes learned by the model, and w_i are the user's attribute weights.

3.2.5 Discussion

Model Interpretability: The technique in BEAMES helps users understand the model output with respect to ground truth, without the need to interpret the internal complexities of the models. This is in contrast to conventional view of model comprehensibility and interpretability, which emphasizes methods to understand the learning of a model by the user [48, 86]. While our work begins to explore this space, there is more work needed to find how users understand and interpret models by observing outputs, helping them in this semi-automatic model selection process.

Why not Automate?: Model selection processes greatly benefit if a model sub-space is discovered from the otherwise large and exhaustive model space, either by inferring user interactions or by direct specifications from users; both of these methods require deploying a semi-automatic model selection approach. Further automated processes bring other challenges. For example, one of the feedback from participants in the user study conducted by Talbot et al. was why not automate everything as opposed to adopting a semi-automatic process [228]? The authors asserted that even though some features in their system could have been automated, there were both interaction and engineering challenges to address if their system was fully automated (i.e., computational costs affecting interactions). As such, automating everything might require exhaustive exploration of an overtly large model space rendering the process computationally expensive and often intractable. On the other hand semi-automatic processes makes it computationally efficient and reasonable to search over a sub-space rather than the overtly large model space often finding acceptable solutions within a reasonable compute time.

User Involvement: While useful, the semi-automatic approach in BEAMES requires a considerable user involvement in the model building pipeline, which has its pros and cons. On the pros side, the user is empowered to pick an optimal model themselves. They inspect models one by one and pick a model which suits their goals. The system guides the user by recommending models to help them discover, which model to review or inspect. On the cons side, it adds extra work on the user's end. They need to manually inspect and review models to find the one that best fits their subjective preferences. It may be less productive if the user is not skilled enough to understand the differences between models or interpret shown model outputs correctly. They might end up selecting a sub-optimal or a completely random model.

Seeking to further ease users' role in model selection and model space navigation, I prototyped Gaggle (described in the next section), a VA system that increases automation in model selection with the goal to simplify usability and complexity of user interactions in multi-model systems.

3.3 Gaggle - interactive navigation of model space

Recent multi-model visual analytics systems make use of multiple machine learning models to better fit the data as opposed to traditional single, pre-defined model systems. However,

while multi-model visual analytic systems can be effective, their added complexity poses usability concerns, as users are required to interact with the parameters of multiple models or inspect multiple models. Further, the advent of various model algorithms and associated hyperparameters creates an exhaustive model space to sample models from. This poses complexity to navigate the model space to find the right model for the data and the task. I prototyped Gaggle, a multi-model visual analytic system that enables users to interactively navigate the model space. In addition, translating user interactions into inferences, Gaggle automatically finds the best model from the high-dimensional model space to support various user tasks. Further, through a quantitative and qualitative user study, I show how this approach helps users to find an optimal (preferred) model for a classification and ranking task.

3.3.1 Overview

In Gaggle, we attempt to solve the problem where the right model to use for a problem is not known a priori. In such cases, one needs to navigate the model space to find a fitting model for the task or the problem. Gaggle enables users to classify and rank data items, where each of these two tasks are supported by finding the appropriate classification and ranking model. To keep the interactivity simple, supported interactions operate on data cases which serve as demonstrations to sample a new model from the model space. For example, users can drag data items into specific classes to record classification task's user preferences. Similarly, users can demonstrate that specific items should be higher or lower within a class by dragging them on top of each other. Gaggle's interactive model space navigation technique applies user feedback by sampling a new set of hyperparameters of a classification and a ranking model every time users interact with the data. Furthermore, based on model performance metrics inferred from user interactions, our technique automatically selects the best model from these candidate models as seen in Figure 25.

In addition, Gaggle addresses a common problem of datasets that either lack adequate ground truth, or do not have it [186, 230, 251]. To resolve this problem, Gaggle allows users to iteratively define classes and add labels. On each iteration, users add labels to data items and then build a classification model. After they finish the classification of data items, users are able to rank data items within each class based on criteria relevant to their task or domain. The need for model space navigation is exemplified in this iterative exploration and model building scenario. During this process, users may change their task definition slightly or learn new information about their data. In these cases, their user feedback may be better modeled by a different model hyperparameterization than their feedback earlier in the process. Updating the class definition or showing better examples directly affects the underlying decision boundary, which the classifier needs to map correctly. For example, in the first iteration, a linear decision boundary might characterize the data. However, when new examples for classes are provided the decision boundary might be better approximated using a polynomial or radial surface (refer Figure 26). In situations like this, our interactive navigation of model space approach benefits the user by finding the correct hyperparameter settings through Gaggle's automatic model selection technique.

3.3.2 User Interface and Interactions

Gaggle's interface has the following main components:
Data Viewer: The main view of Gaggle is the data viewer which shows data items within

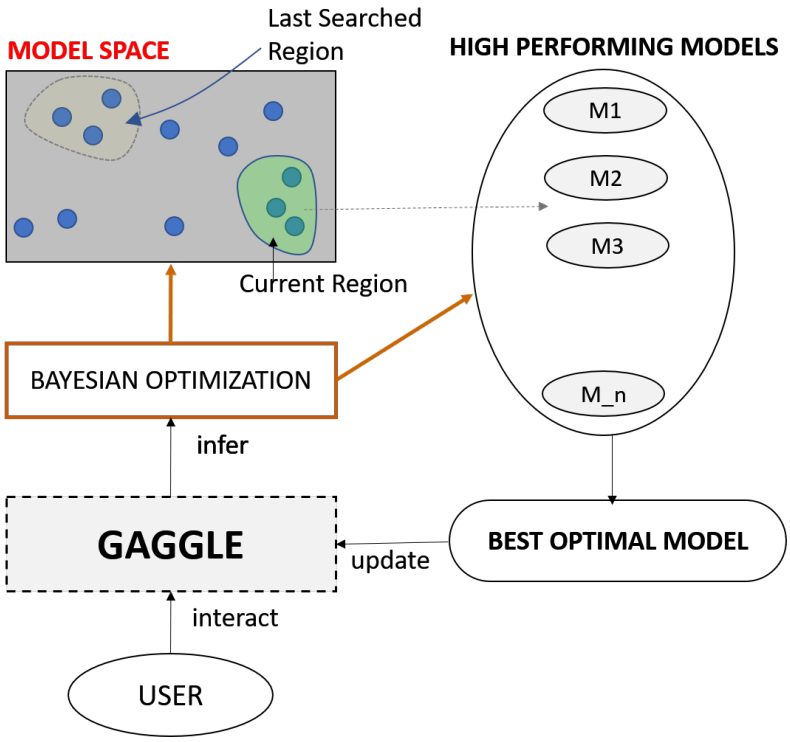


Figure 25: Workflow shown in Gaggle supporting multi-model steering of classification and ranking models.

each class (see Figure 27-(a)). Visually each class is represented with a differently colored rectangular bin. Users can add, remove, or rename classes at any point during data exploration and drag data instances to bins to assign labels (see Figure 27-(i,h)). Further they can re-order instances by dragging them higher or lower within a bin to specify relative ranking order of items. Next, users can trigger Gaggle to construct classification and ranking models. Using a multi-model steering technique (defined later in this chapter) Gaggle finds an optimal model for each model type.

Attribute Viewer: Users can hover over data items in the data viewer to see attribute details on the right (see Figure 28-(a)). Every quantitative attribute is shown as a glyph on a horizontal line. The position of the glyph on the horizontal line shows the value of the attribute in comparison to other data instances (see Figure 27-(b)). The color encodes the data instance's attribute quality in comparison to all other instances (i.e., green, yellow, and red encodes high, mid, and low values respectively).

Data Recommendations: When users drag data instances to a different bin, Gaggle recommends similar data instances which can also be added (see Figure 29). This is to expedite class assignment. The similarity is computed based on the total distance *Da* of each attribute *di* of the moved data instance to other instances in the data.

Interacted Row Visualization : Any interactions user perform on the data viewer is shown on the interacted row visualization (see Figure 27-(c)). It shows data items users interacted with along with any mismatch in its class labels (comparing user-assigned labels as ground truth with predicted labels from a classification model). Similarly, it shows user demonstrated rank order and predicted rank order from a ranking model. The color blue indicates a close match while pink shows a strong mismatch in class label and rank order.

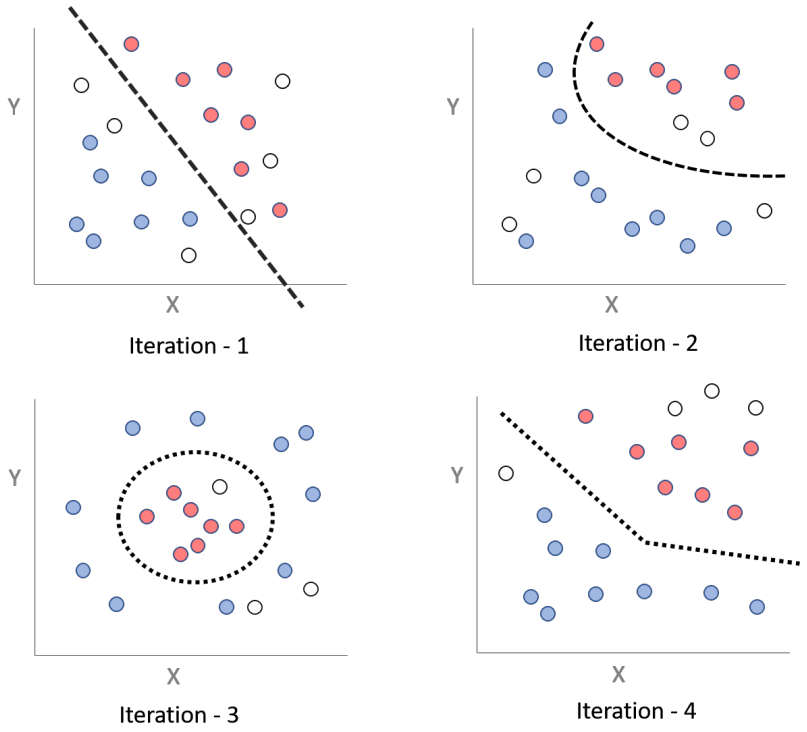


Figure 26: A hypothetical binary classification problem shows how different model hyperparameters may be needed to model the changing user interest at each iteration. Blue and orange points represent positive and negative classes; white points represent data items not interacted with.

Interactions: Gaggle lets users provide feedback to models through a set of interactions explained here:

- Assign Class Labels: Users can reassign classes by dragging data items from one class to another. They can also add or remove classes. These interactions provide constraints to steer the hyperparameters of the classification model.
- Reorder Items within Classes: Users can reorder data items within classes to change their ranking by drag-drop data items (see Figure 27-(g)). This feedback is incorporated as training data for the Ranking model.
- Pin Data Items: When sure of a class assignment of a data item, the user can pin it to the respective class bin. It ensures that data item will always be assigned that class in every subsequent iteration.
- Constrain Classification Model: When satisfied by the classification model, users can constrain the last best classifier. It allows users to move on to show ranking examples.

3.3.3 Usage Scenario

Problem Space: Lets understand how Gaggle can support interactive navigation of the model space. Imagine Jonathan runs a sports camp for baseball players. Being an expert in his field Jonathan knows which features from the data are important to his assessments but also has prior subjective knowledge about the players. In his day to day work, Jonathan

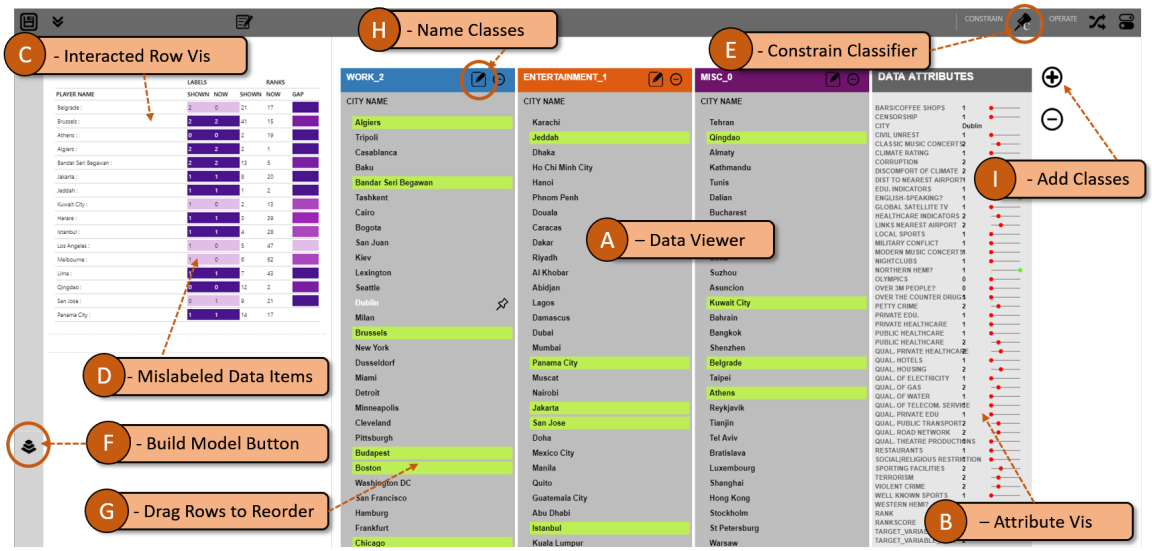


Figure 27: The Gaggle user interface allowing people to interactively navigate a model space to support interactive classification and ranking of data points.

needs to judge areas in which the players need further improvement. He would like to do this by placing players into different categories: "Best Players", "In-form Players" and "Struggling Players".

User-Provided Labeling: Jonathan starts by importing the dataset of baseball players (data publicly available from OpenML [238]). The data contains 400 players (represented as rows) and 17 attributes of both categorical and quantitative types. The dataset does not have any ground truth labels. He sees the list of all the players in the Data Viewer (Figure 29-B). He creates the three classes mentioned above and drags respective players in these bins or classes to add labels. Knowing *Ernie Banks* and *Carl Yastrzemski* as very highly rated players, he places them in the "Best Players" class. Gaggle shows him recommendations of similar players for labeling (Figure 29-A).

Automated Model Generation: Jonathan clicks the build model button from the Side Bar (Figure 27-F). Based on Jonathan's interaction so far, Gaggle constructs the model space comprising of multiple classification and ranking models. Gaggle runs its optimizer to navigate the model space based on Jonathan's interaction to automatically find the best performing model, out of an exhaustive search of over 200 models. When the system responds, Jonathan continues his analysis of the data. He finds player *Ernie Banks* is misclassified and places him in the "In-form Players" class instead of the "Best Players" class. He moves *Ernie Banks* and similar other misclassified players to the correct class label and asks Gaggle to find a model that takes his feedback into account.

Gaggle continues navigating the model space to look for regions where better-performing classifiers and ranking models are more likely. Eventually, it finds a new model and shows Jonathan the updated state of the training data in the data viewer. He reviews the results to find that many of the previously misclassified players are correctly labeled and pins them to ensure they do not change labels in future iterations Next, he looks at the attribute viewer (Figure 27-B) in search of players with high "batting average" and "home runs" values. He moves players that match his criteria into respective labels (e.g., placing *Sam West* and *Bill Madock* in the "In-Form Players" class). After Gaggle responds with a new optimal model,

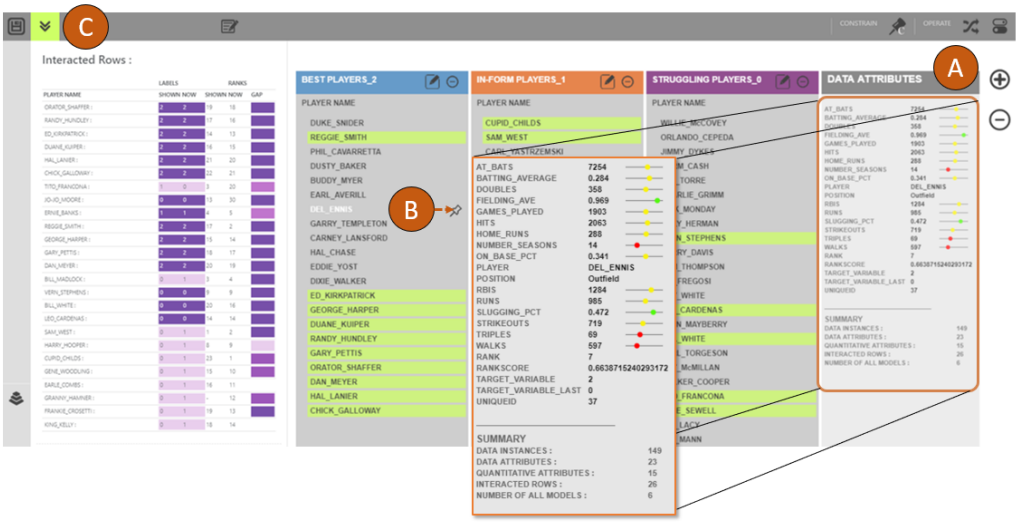


Figure 28: (a) The attribute viewer showing details of each Player on hover. (b) Pin icon, allows pinning a data item to a class bin (c) Set of input control buttons.

he verifies the results returned by the model in the interacted row visualization (Figure 27-C). Content with the model, he accepts the classification model and moves on to rank the players within each class.

For each class, Jonathan drags players up and down to demonstrate his understanding of the ranking of players within classes. He iterates to check the updated optimal ranking model built by Gaggle. He checks the interacted row visualization to find where the model ranked each of the players he interacted with. It shows him expected player rank and assigned player rank (by the current model). He moves player *Norm Cash* and *Walker Cooper* to the top of the "struggling players" class, and moves player *Hal Chase* in the "best players" class down. He iterates further and sees that most of the players are relatively at the correct ranked spot. As a result, Gaggle helped Jonathan navigate the model space to classify and rank players solely based on his prior subjective domain knowledge.

3.3.4 Technique

Bayesian Optimization: To facilitate interactive user feedback and navigation of the model space, Gaggle uses a Bayesian optimization technique [174, 218]. This navigation is initiated by sampling models from the model space as shown in Figure 30. Gaggle seeds the optimization technique by providing: a learning algorithm A, a domain range D_r for each hyperparameter, and the total number of models to sample n for both classification and ranking models. The Bayesian optimization module randomly picks a hyperparameter combination hp_1, hp_2 and hp_3. For example, a model M_1 can be sampled by providing "criteria type" = *gini*, "max-depth" = 30, and "min-samples-leaf" = 12 ("criteria type", "max-depth", and "min-samples-leaf" are the hyperparameters used in Gaggle). Likewise, the Bayesian optimization module samples M_1, M_2, M_3, M_4 ... M_n models. For each such model, it also computes and stores the cross-validation score defined as cv_1,cv_2,cv_3, ... cv_n.

The Bayesian optimization approach uses a Gaussian process to find an expected improvement point in the search space over current observations. For example, a current observation could be mapped to a machine learning model, and its metric for evaluation of

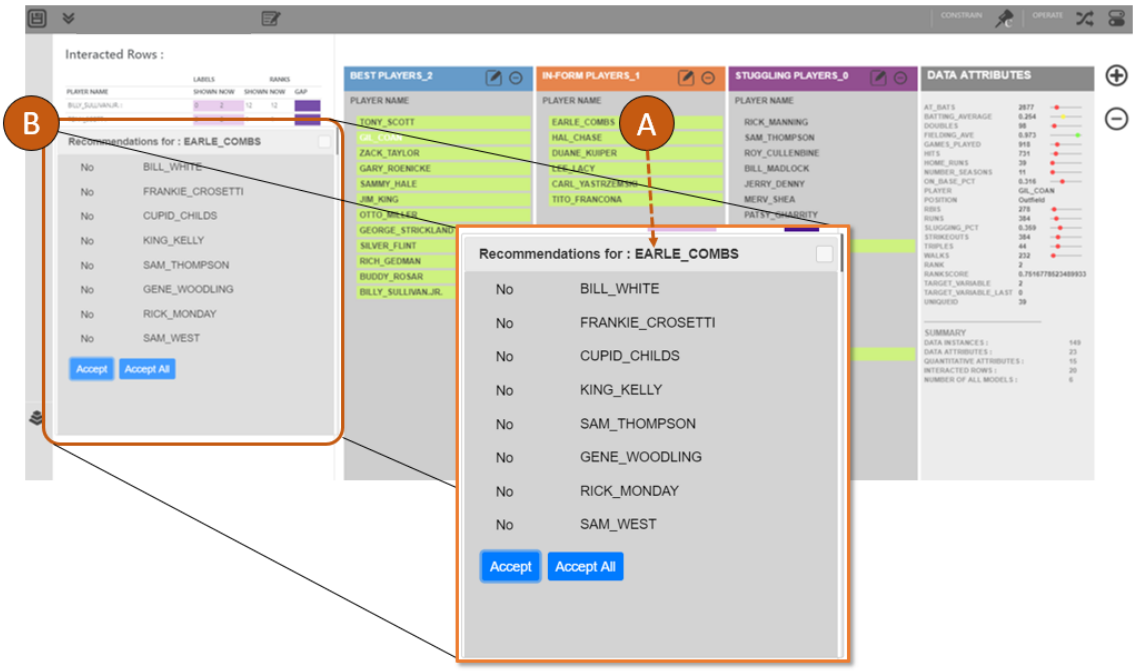


Figure 29: Gaggle's recommendation dialog box.

the expected probability will be the model's cross-validation score. Using this technique, the optimization process ensures consistently better models are sampled by finding regions in the model space where best performing models are more likely to be found (see Figure 25). Next, the Bayesian optimization module finds the model with the best score (see Figure 31). Gaggle performs this process for both classification and ranking models.

Classification Model Technique: Gaggle begins with an unlabeled dataset. As the user interacts with a dataset of n items, labels are added. For example, if the user interacts with e data items, they become part of the training set for the classification model. The rest of the instances $n - e$, are used as a test set to assign labels from the trained model. If e is lower than a threshold value t, then Gaggle automatically finds s similar data instances to the interacted items and places them in the training set along with the interacted data items (s gets the label from the most similar labeled data item in e). The similarity is measured by the cosine distance function using the features of the interacted samples. This ensures that there are enough training samples to train the classification model effectively. As the user iterates and interacts with more data instances, the size of the training set grows, and test set shrinks, helping build a more robust classifier. For each classification model, Gaggle also determines the class probabilities P_{ij}, representing the probability of item i classified into class j.(e.g., $P_{10}, P_{20}, P_{31}, P_{41}, P_{50}, P_{61}, ...etc.$) The class probability is used to augment the ranking computation as they represent the confidence the model has over a data instance to be a member of a said class.

Ranking Model Technique: Gaggle's approach to aid interactive navigation of the model space for the ranking task is inspired by[110, 240]; which helps users to subjectively rank multi-attribute data instances. However, unlike these works, Gaggle constructs the model space using a random forest model (a similar approach to [257]) to classify between pairs of data instances R_i and R_j. While we tested both of these approaches, we adhered to

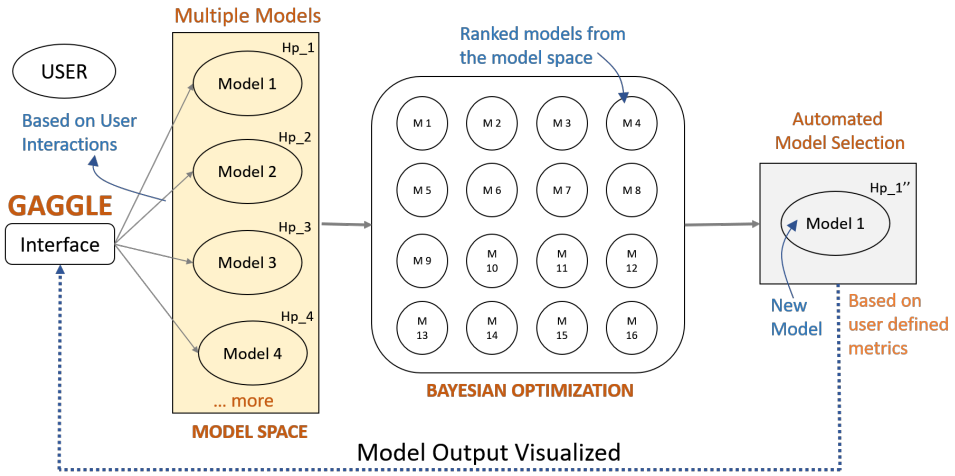


Figure 30: Model space navigation approach using Bayesian optimization to find the best performing model based on user-defined metrics.

random forest models owing to it's better performance with various datasets. Using this technique, a model predicts if R_i should be placed above or below R_j. It continues to follow the same strategy between all the interacted data samples and the rest of the data set. Further, Gaggle augments this ranking with a feature selection method based on the interacted rows. For example, assume a user moves R_i from rank B_i to B_j where $i > j$ (the row is meant to have a higher rank). The feature selection technique checks all the quantitative attributes of R_i, and retrieves $m = 3$ (the value of m is learnt by heuristics and can be adjusted) quantitative attributes Q_1, Q_2, and Q_3 which best represents why R_i should be higher in rank than R_j These features are the ones in which R_i is better than R_j (user can specify higher or lower value of an attribute is better or worse). If $i < j$ (or if the row was meant to have a lower rank), Gaggle again retrieves $m = 3$ features.

We do the same for all the interacted rows, and finally, we get a set of features (F_s, by taking the common features from each individually interacted row) that defines the user's intended ranked order. In this technique, if a feature satisfies one interaction but fails on another, they are left out. Only the common features across interacted items get selected. The set of selected features F_s are then used to build the random forest model for the ranking task. Using the class probabilities (from the classifier) and the ranking models, Gaggle ranks the data instances within each class. A ranking model assigns a ranking score E_{ij} (ith instance, of jth class) to each data instance. A final ranking score is computed by combining the ranking score of a data instance R_i and its class probability P_{ij}, derived from the classification model. It is represented as $R_{ni} = E_{ij} * W_r + P_{ij} * (1 - W_r)$ where R_{ni} is new rank, W_r is the weight of the rank score and $1 - W_r$ is the weight of the classification probability (see Figure 31). Based on the final ranking score R_{ni} the dataset is sorted and presented to the user.

3.4 Gaggle - Evaluation

We conducted a quantitative and qualitative user study to evaluate Gaggle's automatic model space navigation technique to support the classification and ranking tasks. One of the motivation of the study was to get user feedback/responses to Gaggle's system features, interaction design, and workflow. Further, collecting observational data we sought to test

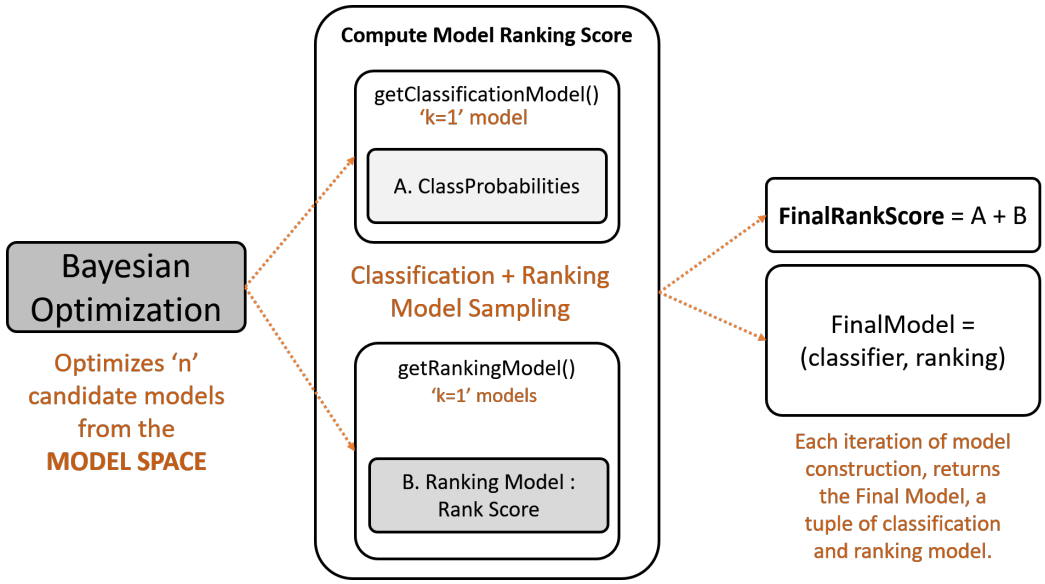


Figure 31: Ranking score computation using both the classification and the ranking model augmented by the bayesian optimization approach in Gaggle.

if our technique helped users in finding an optimal model. We also collected log data and user preference ratings to analyse the data quantitatively.

3.4.1 Participants

We recruited 22 graduate and undergraduate students (14 male, 8 female). The inclusion criteria were that participants should be non-experts in ML, and have adequate knowledge of movies and cities (datasets used for the study). All participants rated themselves fluent in English. None of the participants used Gaggle prior to the study. We compensated the participants with a $10 Amazon gift card. The study was conducted in a lab environment using a laptop with a 17-inch display and a mouse. The full experiment lasted 60-70 minutes.

3.4.2 Study Design

Participants were asked to complete 4 tasks: multi-class classification of data items (3 classes), ranking the classified data items, binary classification of items, and ranking the classified data items. Participants performed the above 4 tasks on 2 datasets, Movies [3] and Cities [2]. To reduce learning and ordering effects, the order of the datasets were randomized. In total, each participant performed 8 tasks, 4 per dataset. We began each study with a practice session to teach users about the workflow and interaction capabilities of Gaggle. During this session, participants performed 4 tasks (printed on a sheet of paper). The tasks took 15 minutes, and they included multi-class classification + ranking, and binary classification + ranking on the Cars dataset [1]. We encouraged participants to ask as many questions as they want to clarify system usability or interaction issues. We proceeded to the experimental sessions only when participants were confident enough to use the system correctly.

In the experimental session, participants were asked to build a multi-class classifier and rank data items within the specified classes. Next they performed a binary classification

and ranking task on the same dataset. Then they repeated the same set of tasks on the other dataset. The movies data had 210 items, with 11 attributes, while the cities dataset had 140 items with 45 attributes. We asked participants to create specific classes for each dataset. For the Movies dataset multi-class labels were *sci-fi*, *horror/thriller*, and *misc*, and *fun-cities*, *work-cities*, and *misc* for the Cities dataset. For the binary classification task, the given labels were *popular* and *unpopular* (for Movies dataset), and *western* and *non-western* (for Cities dataset).

3.4.3 Data Collection and Analysis

We collected subjective feedback and observational data through the study. We encouraged participants to think aloud while they interacted with Gaggle. During the experiment sessions, we observed the participants silently in an unobtrusive way to not interrupt their flow. We audio and video recorded every participant's screen. We collected qualitative feedback through a semi-structured interview comprising of open-ended questions at the end of the study. We asked questions such as: *What were you thinking while using Gaggle to classify data items?*, *What was your experience working with Gaggle?*, etc. Further, after each trial per dataset, we asked participants to fill a questionnaire containing likert scale and true/false type questions. At the end of the study, we saved a set of *.csv* files storing the classified and ranked data.

For quantitative assessment, we primarily rely on log data which stores model hyper-parameters per iteration, class labels of the data items per iteration, the model's learning algorithm, data items users interacted with, etc. We considered five dependent variables for this study: *Model accuracy (classification):* the accuracy of the model in predicting correct labels, *Model accuracy (ranking):* the accuracy of the ranking model *Perceived accuracy (classification):* number of correctly labeled data items (as obtained from the 15 data points we ask users to label), *Perceived accuracy (ranking):* number of correctly ranked data items (as obtained from the questionnaire after each session), and *User preference:* preference rating of the system (see Figure 34) provided by the user as a feedback from the questionnaire.

3.4.4 Quantitative Data Analysis

The study also had a quantitative part, the goal of which was to compare our multi-model based interactive model space navigation technique with a single pre-defined model based approach in VA systems. To further understand how non-experts select models in VA systems, with this quantitative study we intended to understand how the current methods of model selection compare against each other. Specifically, this quantitative study compared a multi-model selection with a single model selection approach to learn the trade-offs and answer: (1) how non-experts select models in VA?, and (2) when one approach is preferable over the other?

We refer the multi-model selection approach as MMS and the single model selection approach as SMS. Both conditions use Gaggle's user interface and interactions to minimize the potential confounds caused by different tools, and isolate the effect of MMS. For the SMS condition, Gaggle used a pre-selected ML model. The hyperparameters for this model were chosen by training the model with a preset target label as the ground truth, using SK-Learn's random search technique [205] to find a hyperparameter combination with the best cross-validation score. This provided the SMS condition with a realistic starting model for

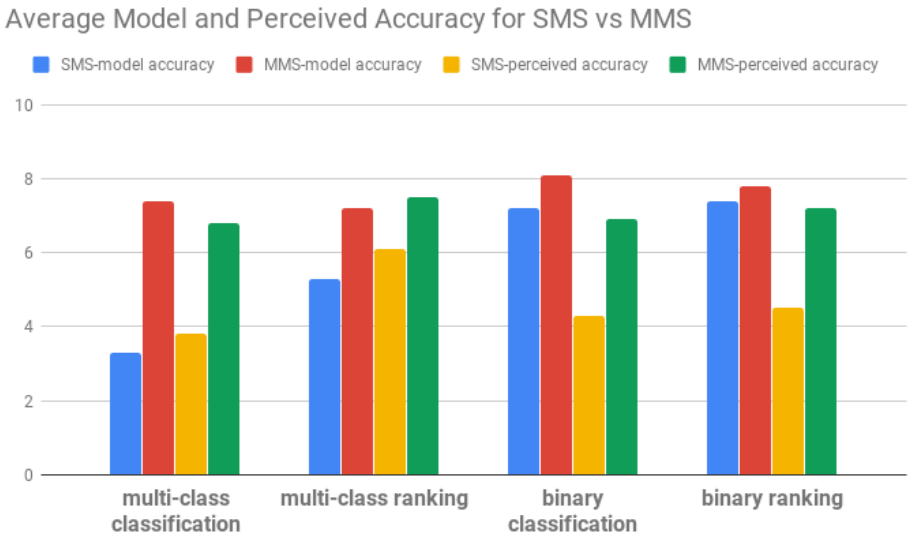


Figure 32: Average Model and Perceived Accuracy of SMS and MMS technique.

the task. The MMS technique performed interactive model space navigation as described in Section 3.3.4. Participants used Gaggle with MMS and SMS condition, one after another to perform the described set of tasks. The order of the systems were randomized to remove any ordering effects.

The primary research questions our study answered were:

Q1 How and when does model switching occur in the MMS condition?

Q2 How does performance compare between the MMS and SMS conditions?

Overall, we hypothesize that MMS will outperform SMS, as it interactively searches through a larger model space in each iteration, and should thus fit the user's preferences more closely. To understand this in more detail, our study tests the following hypotheses:

H1 MMS will outperform SMS (with respect to accuracy) for the combined task of classification (both multi-class and binary) and ranking.

H2 For multi-class classification and ranking problems, MMS will outperform SMS with respect to accuracy.

H3 For binary classification and ranking problems, MMS will outperform SMS with respect to accuracy.

Defining Task Accuracy: We compare SMS and MMS with respect to which one is more capable of building a model that satisfies various user-defined constraints. We analyze and report accuracy for each task (ranking and classification) using the metrics *model accuracy* and *perceived accuracy*. We compared the model accuracy and perceived accuracy for each interface and tested for statistically significant differences. Figure 32 shows model and perceived accuracy for each task.

Task Accuracy Across All Tasks: To test **H1**, we conducted a Friedman Test for Repeated-Measures and found a significant difference in model accuracy between the interface type SMS ($M = 0.512$ $[0.477, 0.547]$) and MMS ($M = 0.786$ $[0.732, 0.840]$) for all four tasks combined. Post-hoc Wilcoxon signed-rank tests with Bonferroni correction

Table 2: The mean, SD, and p-value for all tasks combined for both SMS and MMS. All p-values are Bonferroni-corrected.

Tasks	**Single-Model (SMS)**	**Multi-Model (MMS)**	**p-val**
Model Accuracy All tasks	M = 0.512 SD = 0.035	M = 0.786 SD = 0.054	< 0.05
Perceived Accuracy All Tasks	M = 0.410 SD = 0.004	M = 0.721 SD = 0.013	< 0.05

Table 3: Mean, SD, and p-values of perceived accuracy for classification and ranking using SMS and MMS. All p-values are Bonferroni-corrected.

Tasks (Perceived Accuracy)	**Single-Model (SMS)**	**Multi-Model (MMS)**	**p-val**
Multi-class classification	M = 0.324 SD = 0.121	M = 0.611 SD = 0.100	$p < 0.05$
Multi-class ranking	M = 0.418 SD = 0.055	M = 0.578 SD = 0.210	$p < 0.05$
Binary classification	M = 0.592 SD = 0.101	M = 0.622 SD = 0.051	$= 0.98$
Binary ranking	M = 0.583 SD = 0.087	M = 0.654 SD = 0.212	$= 0.92$

found statistical significance (M($p < 0.05$)). We used the Friedman Test for Repeated-Measures as it is a good indicator of statistical significance for multi-class classifiers with multiple datasets as suggested by[59]. Similarly, we conducted Friedman Test for Repeated-Measures with Post-hoc Wilcoxon signed-rank tests for perceived accuracy for SMS ($M = 0.410$ $[0.406, 0.414]$) and MMS ($M = 0.721$ $[0.708, 0.734]$). We found MMS significantly outperformed SMS for all tasks (M($p < 0.05$)), confirming **H1** (see Table 2).

Task Accuracy for Multi-class Classification: To test **H2**, we conducted a Friedman Test for Repeated-Measures using model accuracy between the two conditions to determine effects specifically on multi-class classification and ranking tasks. The results show that participants performed significantly better with MMS ($M = 0.824$ $[0.821, 0.827]$) than the SMS ($M = 0.623$ $[0.618, 0.628]$) for multi-class classification with M($p < 0.05$). Similarly, results for the multi-class ranking indicate M($p < 0.05$). Then we conduct a Friedman Test for Repeated-Measures between the two interfaces to determine effects on multi-class classification and ranking tasks with respect to perceived accuracy. These results confirm **H2**. See Table 3 for results.

Task Accuracy for Binary Classification: We followed a similar process of analysis for binary classification and ranking tasks. The Friedman Test for Repeated-Measures with post-hoc Wilcoxon signed-rank tests with Bonferroni correction on model accuracy could not prove the statistical significance ($p = 0.73$ for binary classification and $p = 1.03$ for binary ranking task) across SMS and MMS. Similarly we did not observe statistical significance on perceived accuracy ($p = 0.98$ for binary classification, and $p = 1.43$ for binary ranking task). Thus, we cannot conclude that MMS outperformed SMS with respect to model accuracy for binary classification and ranking tasks. These results do not confirm **H3** (see Table 4).

Model Switching Behavior: For all participants, the MMS technique resulted in model

Table 4: Mean, SD, and p-values of model accuracy for both classification and ranking using SMS and MMS. All p-values are Bonferroni-corrected.

Tasks (Model Accuracy)	**Single-Model (SMS)**	**Multi-Model (MMS)**	**p-val**
Multi-class classification	M = 0.623 SD = 0.005	M = 0.824 SD = 0.003	< 0.05
Multi-class ranking	M = 0.781 SD = 0.040	M = 0.912 SD = 0.023	< 0.05
Binary classification	M = 0.725 SD = 0.120	M = 0.810 SD = 0.076	$= 0.73$
Binary ranking	M = = 0.81 SD = 0.034	M = 0.832 SD = 0.233	$= 0.92$

Table 5: The change in hyperparameters in MMS and change in cross validation score per iteration for both SMS and MMS

Iter.	**SMS Score**	**SMS Hyperparam**	**MMS Score**	**MMS Hyperparam**
1	0.76	MaxDepth = 4 Criteria = 'entropy' MinSamples = 10	0.58	MaxDepth = 2 Criteria = 'entropy' MinSamples = 3
2	0.45	MaxDepth = 4 Criteria = 'entropy' MinSamples = 10	0.55	MaxDepth = 12 Criteria = 'gini' MinSamples = 8
3	0.45	MaxDepth = 4 Criteria = 'entropy' MinSamples = 10	0.62	MaxDepth = 22 Criteria = 'entropy' MinSamples = 10
4	0.32	MaxDepth = 4 Criteria = 'entropy' MinSamples = 10	0.68	MaxDepth = 24 Criteria = 'entropy' MinSamples = 10

switching. For participants using the Movies dataset (multi-class classification task) the *max-depth* hyperparameter changed values (ranging from from 3 to 18). Similarly, for the Cities dataset (multi-class classification task) the hyperparameter *Criteria* ranged from *entropy* to *gini*. The *min-samples* hyperparameter varied within the range of 5 to 36 for both datasets. For the binary classification task, *max-depth* ranged from 4 to 9 for both datasets. Also we noticed the *criteria* hyperparameter switching from *gini* to *entropy* for both datasets for the binary classification task.

On average the hyperparameters switched $M = 9.34$ $[7.49, 11.19]$ times to support the multi-class classification and ranking task, while the average change was $M = 5.41$ $[4.89, 5.93]$ for binary classification and ranking task. On the other hand, SMS adhered to the pre-defined hyperparameter setting in each iteration. Though on certain iterations SMS was able to satisfy the majority of the user constraints, however, more frequently SMS failed to satisfy most of the user constraints. For example in the classification task in SMS, the model showed very low cross-validation score (higher is better, see Table 5). This is explainable from the fact that the pre-defined hyperparameter settings in SMS could model the correct decision boundary only on some iterations (as observed from the log-data). On the other hand, for each iteration the MMS technique searched for an optimal model which best characterized the decision boundary of the data. Thus, the decision boundary changed per iteration as the user provided new examples.

Model Performance over time: We compare the model performance in terms of the number of wrongly predicted labels for interacted data instances per iteration. For the multi-class classification task, SMS model output was inconsistent which meant on some iterations the model correctly predicted most of the data items, but the prediction quality may drop in later iterations (see Figure 33). In comparison to SMS, MMS showed a more consistent performance gain, meaning the number of correctly predicted labels improved over time. However, for the binary classification task, the model performance for SMS and MMS were comparable (see Figure 33). The results indicate that binary class labels were relatively easier to predict for SMS even if the underlying model's decision boundary was not the best representation of the actual decision boundary of the data instances.

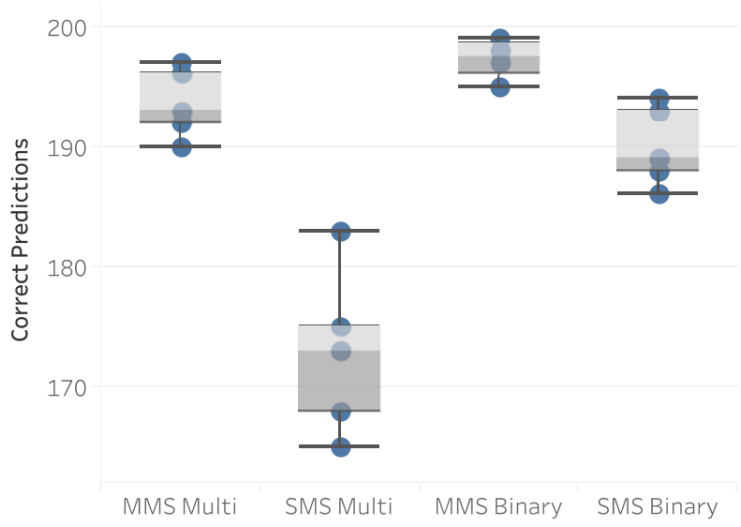


Figure 33: The number of correctly predicted labels for interacted data items.

Number of iterations: Using the SMS technique, participants iterated more (compared to the MMS technique) until they found an acceptable ML model. For example, the average number of iterations and standard deviation for the multi-class classification task on the Movies dataset was $M = 3.85$ $[3.31, 3.49]$ with MMS, compared to $M = 6.36$ $[5.08, 7.64]$ with SMS. Thus compared to MMS, users had to iterate more with SMS to adjust the model behavior by showing meaningful examples to suit their goals.

User Preferences Rating: We collected user preference rating for all four tasks (see Figure 34). The scores were between 1-5 (1 meaning least preferred, 5 meaning highly preferred). We found the user preference rating for multi-class classification (3.97) and binary classification (4.17), task very close to each other. Though in general users approved Gaggle's simplicity to allow them to classify and rank data samples, they seemed to prefer Gaggle for the binary classification and ranking task owing to higher accuracy and consistently matching users interpretation of the data.

3.4.5 Qualitative Feedback

Here we describe the qualitative feedback from the participants.

Drag and drop interaction: All the participants liked the drag and drop interaction to demonstrate examples to the system. *"I like the drag items feature, it feels very natural to*

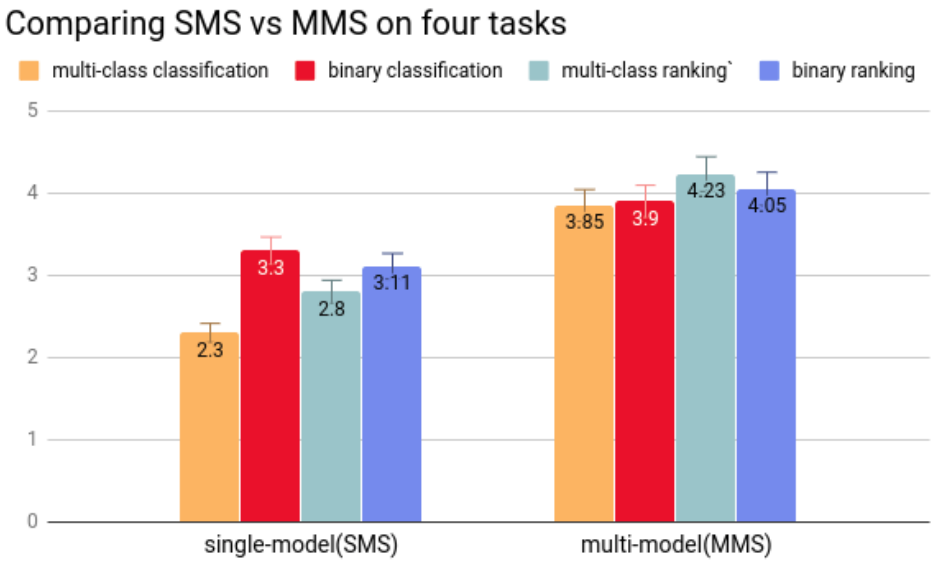


Figure 34: User preference for SMS vs MMS for all four tasks.

move data items around showing the system quickly what I want" (P8). However, with a long list of items in one class, it can become difficult to move single items. One participant suggested, *"I would prefer to select a bunch of data all at once and then drag-drop them as a collection"*.

Need for Filtering: Participants found the design of the attribute viewer helpful to find representative data items to label. *"Seeing the attributes with red, yellow, and green color encoding helped me understand movies which are high or poorly rated .... "* (P14). However, others pointed out that filtering the data by attributes and working on the subset P19 said *"Setting a filter range by attribute would have shown me all the cities with English speaking population ... "* A goal of the current design was to encourage users to think at the data item level, however filtering functionality may improve performance by allowing users to focus on subsets.

Ease of system use: Our design goals included shielding users from the complexity of model building and model selection as much as possible. We focused on designing the system in a way that encourages users to think about their prior knowledge and communicate that to the system. Most participants found the system easy to use. P12 said *"The process is very fluid and interactive. It is simple and easy to learn quickly."* P12 added *"While the topic of classification and ranking models is new to me, I find the workflow and the interaction technique very easy to follow. I can relate to the use case and see how it [Gaggle] can help me explore data in other scenarios."*

Recommended items: Recommending data while dragging items into various labels helped users find correct data items to label. P12 said *"I liked the recommendation feature, which most of the time was accurate to my expectation. However, I would expect something like that for ranking also ...."* P2 added *"I found many examples from the recommendation panel and I did not have to scroll down. I felt it was intelligent to adapt to my already shown examples."*

User-defined Constraints: The interacted row visualization helped users understand the constraints they placed on the classification and ranking models. P14 said *"The interacted row visualization shows me clearly what constraints are met and what did not. I can keep track of the number of blue encodings to know how many are correctly predicted"*. Even though the green highlights in the data viewer also mark the interacted data items, the interacted row view shows a list of all correct or incorrect matches in terms of classification and ranking. P3 remarked *"Without the interacted row view, I cannot keep track of all the data items I interacted with, especially when the number of items I interacted was over 20."*

Labeling Strategy Few participants changed their strategy to label items as they interacted with Gaggle. They expected it might confuse the system. However, to their surprise, Gaggle adapted to the interactions and still satisfied most of the user-defined class definitions. P17 said *In the movies data set, I was classifying sci-fi, and thriller movies differently at first, but later I changed based on recent movies that I saw. I was surprised to see Gaggle still got almost all the expected labels right for non-interacted movies....*

3.5 Summary

This chapter explained model selection and its relevance to finding optimal models for non-experts. Further, I discussed the full spectrum of model selection showing the pros and cons of each approach. Through BEAMES I learned a semi-automatic model selection technique guides users to discover an optimal model, but it may entail significant user involvement to inspect models and their outputs. To combat this problem I prototyped an interactive model space navigation technique using an automated model selection approach in Gaggle to simplify user interactions of multi-model VA systems. Current VA techniques rely on a pre-selected model for a designated task or problem. However, these systems may fail if the selected model does not suit the task or users goals. As a solution, the presented technique in Gaggle helps users to automatically find a model satisfying specified preferences by interactively navigating the high-dimensional model space. Specifically, these approaches are well-suited for situations where there is no (or inadequate) ground truth, the quality of training data is questionable, or the end user knows more about the data than what is explicitly contained in the training set.

CHAPTER IV

EVALUATING MULTI-MODEL STEERING AND MODEL SELECTION WITH DOMAIN EXPERTS

Q2: *How effective are multi-model steering and interactive model selection in supporting domain experts to construct clustering models?*

This chapter describes a domain study with biology researchers. This was conducted to evaluate: (1) multi-model steering and (2) model selection techniques, previously prototyped for classification and regression tasks. In this study we designed an interactive visual clustering system specifically for biology researchers at the Georgia Institute of Technology.

4.1 Problem Statement

Clustering is the task of summarizing and aggregating complex multi-dimensional data in such a way that items in the same group are more similar to each other than those in different groups. As such, clustering has a widespread application in several domains including biology [164], chemistry [62], and social sciences [18]. Domain experts often want to perform clustering to find groups of data items that share common characteristics with respect to data attributes. For example, a biologist who wants to investigate genome data can cluster gene sequential data according to similarity between their expression profiles.

Though clustering plays a pivotal role in biologists' data exploration, it takes non-trivial efforts for biologists to find the best grouping in their data using existing tools. Visual cluster analysis is currently performed either programmatically or through menus and dialogues in many tools, which require parameter adjustments over several steps of trial-and-error. Based on our collaborations with a group of biologists, we found that they use tools like SAS and/or programming languages like R to run cluster analysis on their data. These tools require users to specify clustering algorithms and parameters in written scripts. The absence of user-friendly tools may increase execution costs and impede the adoption of clustering methods for data exploration in addition to being time-consuming, cumbersome, and often error-prone. One of the biologists stated that: *"my process [data exploration process] is sometimes slow. [...] I search for code snippets online. After finding the code, it takes 2 or 3 trials to get the code working."*

Existing visual analytic systems that support visual clustering are often complex, and require careful tuning, selection, and parameterization of the clustering models (e.g., [33, 37, 41, 60, 105, 131, 212]). Interaction complexity in such systems often poses fundamental usability challenges for those domain experts who may not have formal data science training [68]. Furthermore, it is challenging for domain experts to directly apply their knowledge into the clustering processes. For example, biologists exploring genome data might want to merge two clusters because of the similarity of evolutionary history of the genes located in two clusters. Alternatively, they might want to subdivide a specific cluster to estimate the disease risk of genes in different sub-clusters in a specific population. However, many of existing tools do not provide visual guidance on how to reach desirable results or translate users' analytic goals into a proper setting of algorithm and parameter.

4.2 Interactive Visual Clustering

As a solution we designed Geono-Cluster, a novel visual analysis tool designed to support cluster analysis for biologists who do not have formal data science training. Geono-Cluster enables biologists to apply their domain expertise into clustering results by visually demonstrating how their expected clustering outputs should look like with a small sample of data instances. Using multi-model steering and a semi-automatic model selection approach, the system translates user interactions into numerical processes that update the underlying cluster distance functions. Further, the system predicts users' intentions and generates potential clustering results. Using the system, users can create, evaluate, and refine clustering results in order to gradually reach the most optimal clustering result for their analysis goals.

We have developed Geono-Cluster in collaboration with biologists investigating disease risks frequency across different populations. We closely followed the design study protocol [207] to derive system requirements, tasks to be supported, and design guidelines based on feedback from biologists. Instead of requiring biologists to transform their clustering tasks into system specifications by going through layers of menus or programming it, Geono-Cluster allows biologists to directly apply their domain expertise by visually demonstrating how their expected changes should look like (e.g., dragging one cluster and dropping it over another cluster to show their interest in merging the clusters). To evaluate our approach, we then conducted a qualitative study with six expert biologists at the Georgia Institute of Technology. In this evaluation, we observed how our tool helps biologists to cluster their data and identify challenges they encounter while using our tool.

4.3 Background

Researchers have been investigating various techniques and approaches to facilitate interaction in clustering analysis, with the goal of bringing a human in the loop. Effective user interaction is critical to the exploratory data analysis process, and thus to the success of the visual analytic systems for visual clustering. A large body of previous work designed and implemented interactive tools to support interactive visual clustering and analysis (e.g., [19, 37, 56, 58, 66, 92, 100, 131, 146, 162, 197, 208, 245]). Clusterophile [58] and Clusterophile 2 [37] are both designed to enable users to explore different choices of clustering parameters and reason about clustering instances in relation to data dimensions. iVisClustering [134] is another tool that supports document clustering based on a widely used topic modeling method called latent Dirichlet allocation (LDA). Hu et al. [100] and Guo [92] separately developed interactive tools that allow users to select features while clustering their data. ClusterSculptor [162] is another tool that aids data scientists in the derivation of classification hierarchies in cluster analysis.

There exist some visualization tools that are designed for clustering analysis of biological data. StratomeX [139] is an interactive visualization tool that enables users to explore the relationships of sub-set data samples across multiple genomic data types. StratomeX is mainly designed to support tasks with "comparative nature" (e.g., evaluate how well two or more stratifications support each other). CComViz [107] is a different application that uses the parallel sets technique to compare clustering results. Kern et al. proposed novel methods for evaluating and comparing cluster results and implemented their methods into StratomeX [119]. XcluSim [150] is another tool for bioinformatics data helping users to compare multiple clustering results, supporting a diverse set of algorithms. Geono-Cluster differentiates itself from the aforementioned work mainly by supporting biologists'

visual clustering analysis by interaction with data items that do not require specification and inspection of cluster model algorithms, hyperparameters, parameters or performance metrics. Furthermore, unlike existing tools, Geono-Cluster enhances user interaction by enabling users to interactively define clustering results by their demonstration on data items, which is more user-friendly and easy to understand for domain experts.

4.4 Formative Assessment

In this section, we conduct a formative assessment to characterize our domain experts' workflow, derive tasks and requirements from it, and generate design guidelines to design Geono-Cluster.

4.4.1 Characterizing domain experts, data, and tasks

The dataset used by the biologists is from GWAS Catalog [4], which includes published SNPs (single-nucleotide polymorphisms) and association studies. During data analysis, biologists often focus on certain features of their dataset such as, disease/trait, SNP identification number, risk allele frequency, p-value, and odds ratio/beta. Focusing on those values, they try to answer questions like, how and why disease risk frequencies differ across populations, what are the statistical power to detect those known SNPs, and how well associations found in one population can transfer/replicate well to another population. We collaborated with the biologists over 13 months to design and build solutions for supporting interactive visual clustering of disease risk factors. To answer such questions, researchers cluster their data to investigate patterns and relationships of position on the genome, risk allele, and risk allele frequencies that impact diseases risk frequencies across different populations. This is an iterative process and biologists frequently create customized clusters, merge/split clusters, and investigate sub-clusters within a specific cluster to test their hypotheses based on their expertise.

4.4.2 Tasks and Requirements

Following a user-centered method [163], we began our iterative design process by investigating current practices, needs, and challenges. We conducted multiple group discussions with two biologists at the Georgia Institute of Technology. We started our discussions with the biologists by asking them: 1) what kinds of questions do they ask and answer while exploring their data? 2) why do they perform clustering tasks during their analysis? and 3) how do they currently create clusters? Then, we freely continued our conversation that touched upon the tools, analytic methods, and challenges they face during the process. We took notes during all the group discussions. We then read through our notes to gain a better understanding of the requirements and challenges these biologists encounter while clustering their data. Below we describe three commonly performed clustering operations:

T1: Hand-craft, Merge, and Split Clusters: Biologists apply their domain knowledge to create customized clusters to better understand which factor(s) is causing the ascertainment bias on the dataset that are being used popularly. For example, one of the biologist stated: *"Given the identified SNPs [single-nucleotide polymorphisms] that are associated with common disease and traits, it's interesting to create a cluster of SNPs."* In addition, biologists apply their domain expertise to merge or split two or more clusters depending

on how related they think the clusters are based on given feature(s). Another biologist reported that *"In my new project, we are comparing Africans to non-Africans. In this case I merge Americans, East Asians, and Europeans as one cluster, and compare that to Africans data."*

T2: Divide each cluster to sub-clusters: Biologists often investigate sub-clusters within a specific cluster to: 1) understand which other factors can affect the cluster, 2) see trends and patterns in the sub-clusters, with respect to chosen features, and 3) further investigate the risk of disease across subset of data within a cluster.

T3: Adjust feature contributions: Biologists need to easily see by how much different attributes contribute to computing a cluster. Moreover, they often need to adjust the importance of different features used for computing a cluster. Biologists currently have to programmatically iteratively adjust the importance of features, execute the code, and visualize the outcome.

4.4.3 Design Guidelines

We needed to explore alternatives and make design decisions to better support the aforementioned tasks for non-experts. We developed a set of design guidelines to inform those interested in developing visual analytic tools for domain experts (in particular biologists) based on existing systems and our experiences through several design iterations with the biologists.

G1: Translate user interactions (from demonstrations) using multi-model steering and a semi-automatic model selection: Instead of requiring domain experts to specify the clustering models by programming or going through layers of menus, the tool should provide an environment that enables domain experts to visually demonstrate how the expected clustering outcomes should look like. By translating the input user interactions, the system could estimate the user's intention and generate appropriate results using multi-model steering and a semi-automatic model selection that I have deployed before in my research.

G2: Allow user interaction to drive recommendations: As domain experts explore their data, their interests will evolve. Thus, the clustering models should be recommended that adapt to users' analytic goals. To steer the multiple cluster models users can specify their expected visual outcomes by directly manipulating the visual elements representing data items and clusters (e.g., move one or more points from one cluster to another). In addition, users can also directly adjust feature contributions to update the clustering results.

G3: Enhance interpretability of recommendations: Recommended clustering results should be presented in a transparent manner so that users can extract the most contributing features used for clustering results. This requires recommended clustering results to communicate distributions of feature values of members in different clusters clearly.

4.5 Geono-Cluster

4.5.1 User Interface

Geono-Cluster's interface consists of four views: a Cluster View, a Recommendation Panel, a Table View, and an Attribute Panel. See Figure 35 for more details.

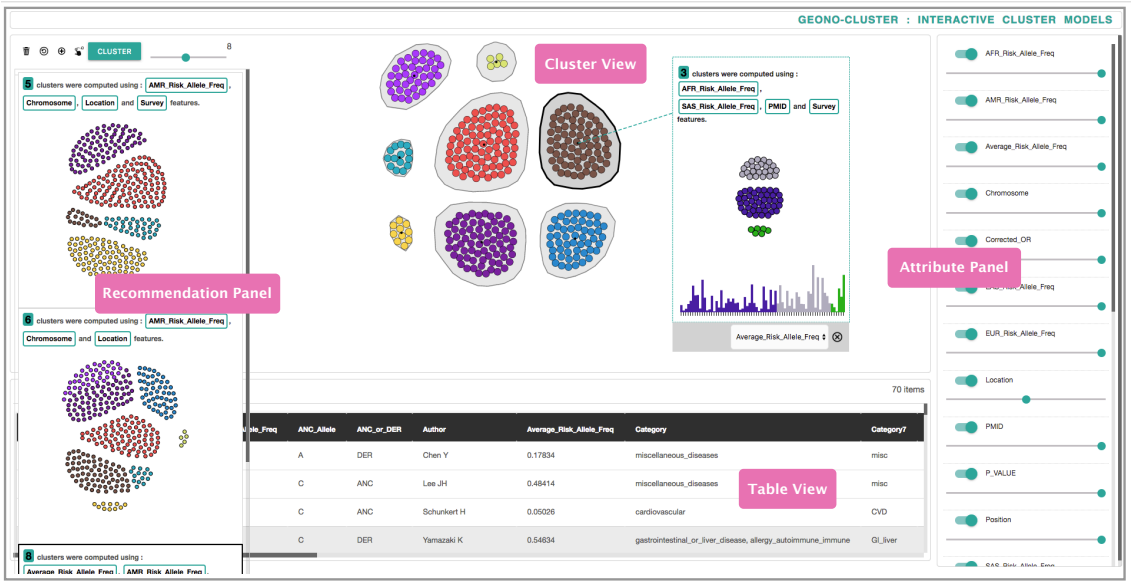


Figure 35: The Geono-Cluster user interface consists of a Cluster View, a Recommendation Panel, a Table View, and an Attribute Panel. Cluster view visualizes the clustered data and provide a medium for users to provide visual demonstrations. Recommendation panel shows different clustering results based on the demonstrations provided by users. Table view shows a tabular representation of the loaded dataset. Attribute Panel lists the attributes of the loaded dataset and their weights.

Cluster View visualizes the clustered data as Figure 35 shows. For testing their hypotheses, biologists often perform actions at the level of data items (e.g., move data items from one cluster to another). We visually present each cluster and its members on the Cluster View. The colored circles in each group represent members of a cluster; the surrounding hull represents the cluster. Users can hover over a circle, which prompts relevant attribute details of the data. Users can specify the number of clusters using the slider shown on the top-left. Cluster View is an environment similar to a spatial workspace in which users can move data items to structure their information and provide visual demonstrations **(G1)**. For example, a biologist might notice a set of data items should not be in a specific cluster. Thus, she can demonstrate that those points belong to a different cluster by dragging them from one cluster to another. The system uses the visual demonstrations provided by the users to steer the underlying recommendation engine **(G2)**.

The Cluster View shows an overview of the clustering results and then encourages users to query additional information (e.g., tooltips, attribute distribution histograms) as they explore the data. The visual representation of the Cluster View powered by a force-directed layout algorithm shows the size and shape of each cluster, the number of clusters, and an overview of clustered data items without overwhelming users with too much information. Furthermore, the design encourages cluster-level interactions such as merging two clusters, or splitting a cluster to refine or customize a cluster. Within each cluster the position of the node (a data item) placed at the center represents a stronger cluster membership than those that are on the periphery. This allows users to understand the clustering probabilities of each data item in relation to others. However, each cluster in the layout is positioned by the force-directed layout simulation that does not capture if two clusters are similar or different. We deliberately restricted ourselves to communicate that information as we

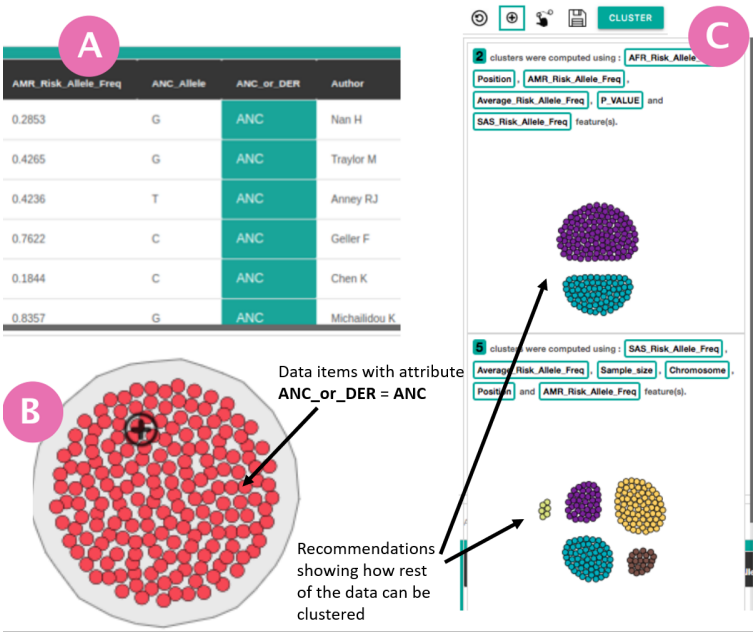


Figure 36: **A** Users can select a subset of data items from the table view, shown with green highlight. **B** The system finds similar data items and places them in one single cluster. **B** Recommendation panel showing how the data set can be clustered using other feature combinations and cluster models based on demonstrated interactions.

intended users to inspect differences between clusters by viewing the thumbnail previews of recommended models.

Recommendation Panel shows different clustering results. Based on users' demonstrations on the Cluster View, the system recommends a set of appropriate clustering outputs (see Figure 35). To compute the recommended clustering results, the underlying recommendation engine takes into account different (1) clustering techniques/algorithms; (2) combinations of attributes/features; and (3) clustering hyperparameters (i.e., varying 'k' for k-means clustering technique). Read section 4.5.3 for more details.

During the design process of Geono-Cluster, we examined different ways of presenting recommended clusters. We first considered showing all the recommended clustering results as small thumbnails in the Recommendation Panel. The biologists liked the idea and the way that we recommended clustering results. However, the main challenge that biologists encountered was that they were not able to infer detailed information from the small thumbnails. Thus, they requested adding textual description of details about each clustering result in the recommendation. Currently, each thumbnail includes a textual description about the number of clusters, features used to compute the clustering recommendations, and a visualization of the clustering result showing distribution of data items over clusters **(G3)**.

Initially, we designed the recommendation module to update the view with new clustering recommendations whenever users show their demonstrations and/or adjust feature contributions. However, our users revealed that such approaches may distract their ongoing investigations on the current results. Thus, we compute cluster recommendations in the background but do not show the results immediately. Once the computation is done, a notification pops up, encouraging users to explore the results on demand by toggling the

'show recommendations' button (see Figure 36-C).

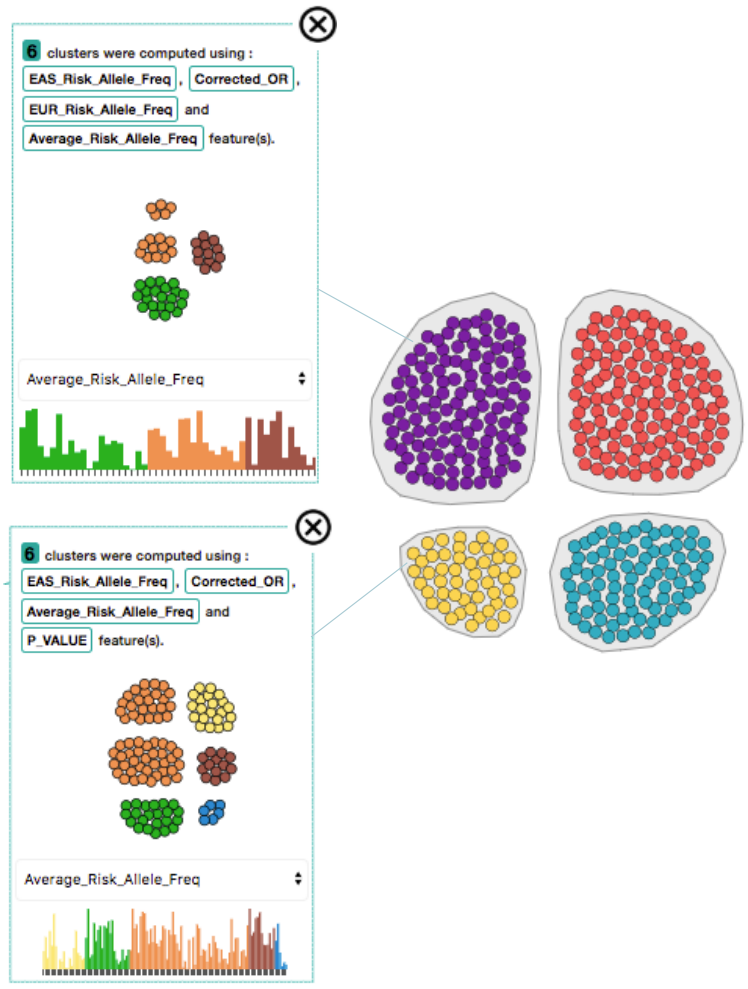


Figure 37: Megan clicks on the "+" icon to open the sub-cluster panel for clusters 1 and 4. Bar chart views showing comparisons of the feature Average-Risk-Allele between Cluster 1 and 4.

Table View shows a tabular representation of the loaded dataset where each row is a data item (see Figure 35). The initial version of Geono-Cluster did not include the Table View. However, biologists requested adding this view since it enabled them to check the raw data. The table updates on user's interaction on Cluster View. For example, selecting the hull of a specific cluster updates the Table View to show the details of items in the selected cluster. Users can click on a cell to find similar rows whose value are similar to the value of the item in that cell. This operation works on both quantitative and categorical data types. For categorical and ordinal data attributes we consider two data items similar if there is an exact match between them. For the numerical variables we define a threshold to measure similarity between two values. This technique allows users to filter and select a subset of data instances (enables selection of multiple rows simultaneously, see Figure 36-A).

Attribute Panel lists the attributes of the loaded data set as Figure 35 shows. Users can turn on and off a set of attributes which directly affects the clustering algorithm. Furthermore, users can also adjust attribute contributions, specifying relative importance

of the selected attributes to define cluster memberships **(G2)**.

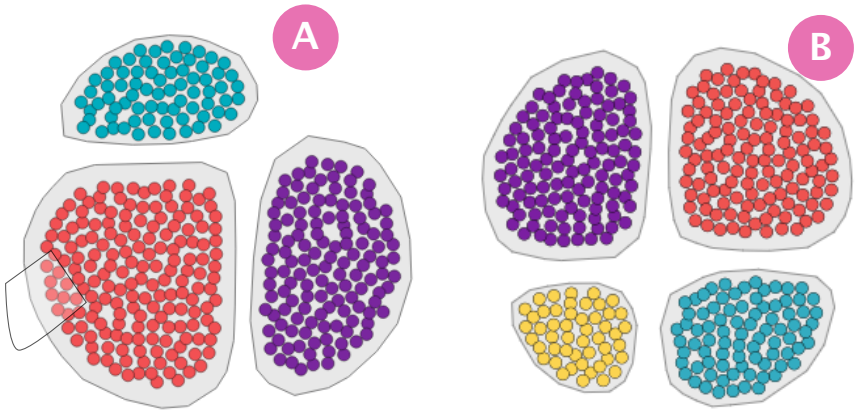


Figure 38: A Lasso selection **B** The system finds similar data items and places them in one single cluster. **B** Recommendation panel showing various ways the data set can be clustered using other feature combinations and cluster models.

4.5.2 Interactions

In this section we discuss how Geono-Cluster supports interactive operations commonly performed by biologists.

Merging and Splitting Clusters (T1): Users can merge two clusters by drag-and-dropping the cluster on top of another cluster. Further they can drag a point out of the cluster and drop into either i) another cluster or ii) blank space. Drag-and-drop items into blank space is translated as forming a new cluster of the selected items outside the current cluster (see Figure 36-B). Demonstration-based cluster customization enables users to interact with the data directly and removes any mid-level instruments such as control panels or menus.

The merge interaction is derived from the previous work by Sarvghad et al. [203], in which they enabled HIV researchers to merge bars in bar charts by dragging one bar and dropping it over another bar. Biologists liked this interaction design and found it "direct and intuitive". To Split clusters, we initially enabled biologists to select the data items by clicking on each circle representing a data item. However, biologists found it cumbersome and time-consuming. So, we implemented the lasso-selection so that users can select multiple data items easily (see Figure 38-A). This operation allows user to brush over a set of data samples (represented as circles) in the Cluster view. In response the system extracts those samples from the current cluster and places them in a new cluster. If data samples from multiple clusters are selected (using lasso selection), then the system makes a new cluster from these lasso picked data samples.

Sub Clustering (T2): Hovering over a cluster reveals a plus button. Users can click on it to open a sub-cluster panel on the cluster view, which shows subgroups of the data items within the selected cluster. In addition, a bar chart shows the distribution of a chosen attribute. Alongside, text description highlights the attributes that were used to compute the sub-clusters. Given that the users are non experts in data science, we do not present the quality metrics (e.g., silhouette scores and homogeneity score). Instead, we describe

cluster models by showing thumbnail previews of clustering results with text descriptions as Figure 37 shows.

Delete data items or Clusters (T3): Our discussion with biologists revealed that they sometimes need to 'exclude' data items or clusters from their analysis while testing a hypothesis. Using the delete button (drag-drop data samples or clusters on the delete button) they can remove subset data items from the Cluster View.
Creating Customized Clusters: Users can select a subset of data items by clicking on the rows shown on Table View (each row represents a data item). After selecting a subset of rows, users can drag-and-drop them on the cluster view to demonstrate their interest in creating a cluster view, in which all the selected data items fall in the same cluster. Users can iteratively repeat the process, and each drag-and-drop operation forms a new cluster in the Cluster View.

4.5.3 Computational Techniques

This section describes the underlying computational techniques which enable Geono-Cluster to recommend cluster models by incrementally steering (multiple cluster models) them to adhere to demonstrated user preferences. Our cluster model recommendation process includes the human in the loop. On a high level, the user shows their intentions on a cluster layout. Based on the operations, Geono-Cluster models multiple cluster algorithms and finds top k closest cluster models to the users' intention. Then, the user can refine the results through a series of customizations (instrumented through the interactions described above). In response, Geono-Cluster automatically finds close variants of cluster models and updates the recommendations in the Recommendation Panel. In summary, the system finds a set of cluster models with a distance function that reflects user-demonstrated cluster assignments.

Multiple clustering models: The clustering task begins when the user requests a new cluster layout (when they press the cluster button in the interface). In response, Geono-Cluster generates multiple clustering models M. Each cluster model M_i in M (M_1, M_2, M_3, M_4, ... M_T) is defined by a careful combination of a learning algorithm ω_i and a set of p hyperparameters ϕ, defined as ϕ_{i1}, ϕ_{i2}, ϕ_{i3}, ϕ_{i4}, ... ϕ_{ip}. Applied clustering algorithms include K-Means, DBScan, Agglomerative Clustering, and Spectral Clustering. In the evaluation of the system, we used K-Means cluster model as we observed through our design-study that most of our users are familiar with packages in "R" to use K-Means clustering (with default parameterization) to cluster the genome data. However, depending on the need of the user and the data used, Geono-Cluster can be extended to use other clustering methods. Nevertheless, each algorithm has its hyperparameters. For example, K-Means is a learning algorithm with "k" and the "max-iteration" value as an input hyperparameter. Furthermore, each model M_i in M is assigned a metric score S_i to compute S, which defines the quality of the clustering output (a higher S_i means a better cluster definition). Geono-Cluster uses Scikit-Learn's ML package to construct and evaluate the cluster models using various quality metrics (e.g., Silhouette Coefficient, Davies-Bouldin index).

Recommendation Technique: Geono-Cluster ranks the models in M by their scores S explained below, and visualizes the best clustering layout in the Cluster View. Further, the system allows the user to inspect top f best cluster models from the ranked models M, through the Recommendation Panel (see Figure 35-a). If a user makes any customization to

the shown cluster model M_c (e.g., merge or split clusters), the system automatically updates the recommendations by computing a new set of M cluster models, except the model M_c, which is currently shown in the Cluster View. Per iteration, the system updates S and the ranking of the models M based on user interactions with the data. Next it visualizes the best model in M in the Cluster View and shows thumbnail previews of the top f models in the Recommendation Panel (**G2**).

Geono-Cluster's model recommendation finds the closest fitting cluster assignments, whenever the user customizes the current cluster layout in the Cluster View. However, there can be scenarios that no cluster recommendation matches the user's intended changes. This may occur when users seek clustering results, which are mathematically infeasible. There could be various reasons for it, such as, users may have a different understanding of the data than what the data actually contains, or the data may have noise, etc. In such cases, users may need to be educated to understand the reasons for a different clustering result, which we plan to integrate in the workflow in the future. Currently, in such cases, Geono-Cluster still responds with the nearest best clustering output, though it may not resemble the layout shown by the user. Furthermore, to ensure users can see unexpected clustering results (to ideate and explore), every few iterations the system also recommends a set of cluster models that are randomly parameterized and thus clusters the data in an unexpected way. While our approach may seem similar to active learning (AL) [213] as in both, users specify feedback to the input data that drives the generation of a model. However, in our case the data does not have class labels. Furthermore, in AL, the system "asks" users to give feedback on specific data points, while in our technique users have the freedom to interactively explore and provide feedback any time along the process.

Clustering Metric: Initially the system does not have cluster assignments or labels for any of the data instances. Thus to compute S the initial cluster models M, are evaluated using the Silhouette Score metric [157]. This metric is computed using the mean intra-cluster distance, and the mean nearest-cluster distance for every data instance. As users interact and assign clusters to a set of data instances I, Geono-Cluster applies two types of metrics to calculate S. To compute the first metric S_1, the system finds all the correlational features fc_k and non-correlational features $fc_k^{'}$ that describes each cluster ($k = 0$ to g clusters). Here, fc_k and $fc_k^{'}$ defines how the user characterises each cluster. Next, when M is computed the system computes the correlational features fc_{ik} and $fc_{ik}^{'}$. The system compares fc_k with fc_{ik} (and $fc_k^{'}$ with $fc_{ik}^{'}$) for each model in M to derive the clustering metric S_1, that describes how closely the cluster model M_i adheres to the clusters defined by the user (S_1 is normalized between $0-1$, higher is a better model). The second metric S_2 is based on the labels assigned to data items by users for I. The system finds other data instances $J = N - I$ (N is all data instances) that are similar to the user interacted data instances using cosine similarity distance metric based on their attribute values (categorical variables are one-hot encoded). We apply a similarity threshold β to find a satisfactory number of similar data instances, the value of which is set empirically with multiple trials on the GWAS data. The current prototype does not allow users to interactively control this threshold. However, in future on users request it can be interactively specified using a slider widget. The system automatically assigns to these similar data instances (J) the same class assignments that the users assigned to I. Next using this labelled data, the system applies Homogeneity index score [157] to compute S_2. This metric uses true labels and assigned labels by the system (when a cluster model M_i is applied to the data) to give a score to each model in M. The final score S is defined as the weighted linear combination:

$S = \lambda_1 * S_1 + \lambda_2 * S_2$. Here the weights λ_1 and λ_2 are hyperparameters that are assigned based on how well the clustering outputs satisfied the biologists' expectations.

User driven feature selection: A cluster model M_i is driven by a set of features $F = f_{i1}$, f_{i2},f_{i3},f_{i4} ..... f_{ik} as input to compute the distance function which assigns a set of data items D to individual clusters C. In Geono-Cluster, the set of features F is either computed using feature selection methods e.g., "select K Best" [173], "PCA" [152] or can be retrieved from users if they specify a set of features and their relative weights (from the Attribute Panel supporting the task **T3**). When users specify a set of k features $F_u = f_{i1}$, f_{i2},f_{i3},f_{i4} ... f_{ik} with respective weights for each feature ($W_u = w_{i1}$, w_{i2},w_{i3},w_{i4} ... w_{ik}, the system updates the distance function in the clustering algorithm. The distance function is represented as $\sum_{i=1}^{k}\sum_{j=1}^{n}\left\|x_i^{(j)} * w_i^{(j)} - c_j\right\|^2$, where c_j, is the jth cluster centroid and $w_i^{(j)}$ is the user assigned feature weight.

Sub Clustering: When triggered by users, the system builds a sub-cluster model M_{si}, for data instances E, member of a selected cluster C_i. Unlike the set of main cluster models M, only a single sub-cluster model is generated per cluster (**T2**). For sub-clustering we relied on the parameterization of the best-recommended cluster model for the entire data i.e, best-found parameterization of the K-Means cluster model. To avoid further compute times that may impact real-time interactions, we did not construct and test multiple cluster models for sub-clustering. However, clicking on the "add subcluster" button again for the same selected cluster C_i, the system recomputes the sub-cluster model M_{si}, by randomly choosing a new set of a learning algorithm ω and hyperparameters ϕ; e.g., it picks a new "k" on the "K-Means" cluster model. This technique allows users to rapidly browse a large set of sub-cluster models.

Similar item selection: Users click on a cell (q_j) of a quantitative attribute on the Table View to select a value v_j of the data item d_i. Geono-Cluster finds a set of r data instances, $U = d_a$,d_b,d_c ... d_r, each of whose value v_j falls within a threshold range, say $[+eps, -eps]$. The parameter eps is set for each quantitative attribute Q by heuristics and can be adjusted. This technique allows users to pick data instances which are similar, based on the selected quantitative attribute q_j. Further, users can select another quantitative attribute cell q_k. Next, from the set of selected data instances U, the system finds all instances V which fall within a threshold range of the value selected for attribute q_k. Here the size of V is less than that of U. This technique allow users to filter and select a subset of data instances V from the Table View. For categorical features X, Geono-Cluster performs exact feature value matching instead of matching data items based on a predefined range. Users can drag-drop these V data items to the Cluster View as a single cluster ($C = C_1$). They can continue selecting another set of data items, then add them to the cluster view as a new cluster ($C = C_1, C_2$). Users complete the data exploration or they can request the system to find a model M_i iteratively (**T3**).

Scalability: Unsupervised learning is expensive. As the number of data items increases, the cost of cluster assignments also grows higher. Geono-cluster can run cluster computations for approximately 3000 data items without major delays. For the scope of our study, the number of data items seem practical.

4.6 Evaluation

To evaluate Geono-Cluster, we performed a qualitative assessment with six biologists to collect subjective feedback and observational data. Our study had two main goals: (1) collect qualitative feedback on Geono-Cluster's features and design, and (2) observe how experts perform visual clustering analysis using Geono-Cluster. In particular, our study indicates how Geono-Cluster helps domain experts gain insights into data by interactively building clusters.

4.6.1 Participants and Setting

We recruited 6 biologists (4 male and 2 female). They had $1-2$ years of experiences working with *Gene* related datasets (all with graduate degrees related to Biology, Bio-Statistics or Bio-Informatics). They had not participated in our preliminary evaluation of Geono-Cluster and were also not involved in the design of Geono-Cluster. All participants were familiar with the concept of data clustering and had previous experience with data grouping with at least one data analysis tool (e.g., SAS, R, etc.). Further, as they had previously worked with *GWAS* catalogue data, they were familiar with all the data attributes in the dataset. During the entire study participants used a computer with 17-inch screen and used a mouse to interact with the system. The study took approximately 50 minutes and we rewarded each participant with a $ 20 Amazon gift card.

4.6.2 Procedure

Introduction and Training: Participants were briefed about the purpose of the study and their rights. After filling out the study consent form and a questionnaire on demographics, we asked participants to watch a tutorial video of Geono-Cluster. The video walked the participants through different features and interactions provided by the tool. After watching the video, we asked participants to work with the tool for 10 minutes. In addition, we encouraged the participants to ask as many questions as they want during this stage.

Main Study: The participants were asked to explore the *GWAS* Cataloge [4] data that includes published *SNPs* and association studies. In particular, we asked the participants to imagine their colleagues asked them to analyze the dataset using the visualization tool for 30 minutes and report their findings. Participants were instructed to verbalize analytical questions they have about the data, the tasks they perform to answer those questions, and their answers to those questions in a think-aloud manner. In addition, we instructed them to come up with data-driven findings rather than making preconceived assumptions about the data. The interviewer facilitated participants' verbal reports by asking questions like "what are you trying to do?", "what are your thoughts now?", "what do you think about current groupings?". We tried to avoid interrupting the participants as much as possible during their data exploration process. However, we sometimes reminded that this is a think-aloud study and they need to verbalize their thoughts.

Follow-up Interview: After each participant complete the task, the experimenters asked the participants open-ended questions such as *Tell me about your experience with this tool?*, *What did you like or dislike about this tool?*, *Did this tool help you in your data exploration? If so how?*, *Is there anything else that you want to add?*, *What are the most interesting things you found from data? How did the tool help you discover such findings?*, and *What*

are the major obstacles/roadblocks while using the tool to solve your problems? How did you go around (resolve) the issue? Do you have ideas to improve the tool? Finally, the experimenter thanked the participants who received a $ 20 value Amazon gift card.

4.6.3 Data Collection and Method of Analysis

We screen- and audio-recorded the whole study. During the main study, the experimenter took notes while participants interact with the system. We also collected feedback from a semi-structured interview with open-ended questions at the end of the study. We analyzed around 300 minutes of screen-capture videos from six participants. First, one of the authors transcribed the audio recording of the study. Then, two coders (first and second authors) read the transcribed data (including the think-aloud sessions and the interview responses) to parse a set of meaningful text snippets. After reading the data, each of the coders independently assigned codes (a word or phrase) to best describe the text snippets. Finally we consolidated the codes from the two authors by focusing on the aspects of the responses which highlighted positive or negative feedback with respect to *usability of the system*, *easy of use*, *learning curve*, *future feature requests* or *strategies pertaining to exploratory data analysis using clustering models*. In the following section, we use **P1** to **P6** to respectively denote the participants one to six who participated in the evaluation.

4.6.4 Results and Feedback

Overall, all participants found Geono-Cluster easy to use and effective in performing cluster analysis tasks. However, a few of them experienced difficulties in interpreting the recommendations made by the system. Below, we categorize and discuss the findings of our qualitative study in more details.

System usability: All participants found Geono-cluster's workflow easy to use, intuitive, and engaging. P2 remarked *"I can keep trying new ideas to quickly test different ways to cluster this data."* P4 said *"It's so easy to use, I can quickly iterate and learn about the data much faster, than using packages in R to cluster data."* Further, many other participants found visualization to be a very good medium to learn about the data by exploring different clustering results. P5 said *"I never knew that I can use visual methods to explore clustering result. Currently I use R to cluster my data, then export a CSV file to my team-mates."*

Consistency with user mental model: Participants found the design and workflow of Geono-Cluster consistent with their mental model and expectations. In particular, participants found that it is intuitive to visually demonstrate tasks such as creating, merging, and splitting clusters by demonstration. For example, P3 mentioned: *"it feels intuitive to merge clusters by dragging and dropping one cluster over another one. [...] this is what I would expect to happen."* P5 stated: *"I liked the idea of creating a cluster of items by moving the data items from this table to the empty space [dragging the data items from the Table View and dropping them on the Cluster View to create a cluster]."*

Perceived control over data analysis process: While using Geono-Cluster, P1, P4, and P5 commented on their level of control over the data analysis that resulted from their freedom in interacting with visualizations instead of going through layers of menu items. For example, P1 mentioned: *"This is great because I can construct my own cluster and*

tell the system how I want my clustering outcome looks like." P4 stated: *"It is a powerful idea to enable analysts to use their knowledge about the data items to interactively create clusters. I specifically like how this allows merging and splitting clusters."* The level of interaction directness [20] with the visual representation contributes towards increasing the perceived control of the participants over the data analysis process.

Difficulty in splitting a cluster: Overall, participants found the lasso interaction intuitive and easy to use. However, with lasso selection participants were not very exact about the data items that they wanted to select. For example, after selecting a subset of data items, P3 noted: *"It is hard to be exact with this selection. I don't want this specific point to be selected."* In such cases, participants had to either deselect the items that were selected incorrectly by clicking on them or try to lasso select again. Going forward, we envision designing advanced interaction techniques for easier selection of data items that are located in a close distance from one another.

Interpretability of recommendations: While using Geono-Cluster, participants were sometimes unsure why the system suggested specific recommendations. While designing Geono-Cluster, we decided to show each recommendation as a thumbnail that uses the natural language to explain the most representative features used for the clustering. In addition, each recommendations shows the resulting outcome of the recommended cluster. Although some participants liked how the recommendations were presented, two participants could not immediately understand why specific recommendations were suggested. For example, P2 mentioned: *"I understand what each cluster represents which is good, but I am not sure why these recommendations."* and P3 stated: *"I am curious how these recommendations are added."*. Going forward, we suggest systems to explore design alternatives to explain the reasoning behind recommendations. In situations when the system does not find any cluster recommendations that matches user's demonstrated changes, Geono-cluster shows the nearest best clustering layout. In the future, we are thinking of explicitly communicating this conflict in textual description.

Custom labeling and annotation feature: Two participants found that with growing number of clusters, it became hard for them to remember what each cluster represents and how specific data items became part of a cluster. Thus, they suggested adding a feature that enables them to annotate and label the clusters and to record the operations performed on clusters beforehand. For example, P6 said *"it will be nice to know what each cluster represents, meaning every cluster should have an annotation, or users can add custom annotations. May be label the cluster by the most prominent feature of the cluster."* P3 mentioned: *"Is there a way to label each cluster?"*

4.7 Observations

Our user study reveals that participants usually began exploring the data by framing a hypothesis, asking the questions they want to know, and then performing a set of tasks (as described in section 4.4.2) through Geono-cluster's interface to find the answers. Interestingly, we observed that participants often took two different approaches to perform visual data clustering: **Top-down** and **Bottom-up**. Below, we describe each approach in more details.

4.7.1 Top-down Visual Data Clustering Approach

P1 started their data analysis process by asking *"How does the gene samples differ in disease risk factor by regions and chromosome factors?"* To that end, P1 clustered data items by selecting a set of features from the Attribute Panel and then pressed the *Cluster* button. Next, he checked the recommended cluster layouts from the Recommendation Panel to explore other clustering results based on another set of features. In response, he updated the list of features to cluster the data by and triggered Geono-Cluster to generate a new cluster layout. P4 also followed the same approach; however, he did not have any question to begin with. He initialized the process by pressing the cluster button to start with an initial clustering. Next, he hovered over data items in each cluster to familiarize himself with the data items and find similarity or dissimilarity. He also checked the Table View to compare different data items from various clusters. If the clusters did not match his mental model, he would adjust the features from the Attribute panel. He would then preview the recommended clustering options to further explore a wide range of cluster outputs. This process continued until he was satisfied with the clusters and had a better sense of the data.

A main point here is that in the top-down approach participants mostly avoided interaction at the data item level, but instead they dealt with the full range of features from the Attribute Panel. P1 also verified this point by saying: *"I relied on cluster button to cluster the data, as I do not specifically know much about the data items, so did not use the table's drag-drop feature. Similarly, I did not customize the clusters by using lasso or drag-drop feature initially. I rather re-computed the clusters based on a new set of features that I specify."* However, P1 later confirmed that over iterations when he was more confident about the data, he started using the split and merge operations to customize shown clusters.

4.7.2 Bottom-up Visual Data Clustering Approach

Remaining participants (P2, P3, P5, and P6) followed the Bottom-up approach, in which they mainly relied on interaction at the data item level. They first created a customized cluster by dragging data items from the Table View and dropping them on the Main view as opposed to relying on the cluster button. These participants often interacted with data items to demonstrate their expected outcome.

P2 started her clustering analysis by asking *"How does the gene samples derived from humans/monkeys (ANC) vary from gene samples derived from mixing humans and monkeys (DER) with respect to various diseases?"*. To answer the question, P2 placed all the *ANC* gene samples into one cluster and a few *DER* gene samples into another cluster from the Table View. P2 remarked: *"my strategy is to select a set of data points [items] based on the gene's ancestry, then drag-drop to create a cluster"*. P2 then previewed the recommendations to explore other options to cluster the data based on his specification of clusters. In this process, P2 did merge/split clusters to test different ideas to cluster the data using the lasso-selection and the cluster drag-drop feature. P2 said: *"I also rely on the lasso tool to define other clusters from this, if the cluster appears too big"*.

P6 also followed the same approach. P6 mentioned: *"I want to know if the gene with chromosome factor higher than 6 sampled from America, have higher cancer risk factor? To seek an answer, I find the Table View's data item selection feature quite useful, as I can define my own clusters based on chromosome value or the region the gene was sampled from."* P6 checked the recommendations, however in some cases, P6 did not agree with the recommendations or the features that were used to derive the results. To provide his

feedback for updated results, he customized the best-perceived cluster layout by splitting the existing clusters using the lasso tool and merging smaller clusters into one.

4.7.3 Commonalities in Visual Clustering Approaches

Despite differences in the two clustering approaches discussed above, there were commonalities in how participants clustered their data. In both approaches, participants relied on the cluster recommendations to explore alternative approaches to cluster or learn about the features in the data that affected the cluster definitions. Furthermore, in both approaches, users found customizing clusters very helpful. Operations such as lasso-selection and merge were instrumental in expressing their intentions to steer cluster models. For instance, P3 said: *"My strategy is to compare different populations by Region and compare their risk-allele-freq. To do that, I often split/merge clusters to show the system how I want the clusters to look like."*

4.7.4 Discussion

Model Feedback and Interpretation: Periodic discussion and informal inputs from the biologists clarified that model interpretation and feedback (to the model) is of critical value to them. For example, when Geono-Cluster shows a set of clustering recommendations, users may need to know how they differ from each other, or what logic was implanted to define the displayed clusters. There are many ways to explain this to the user; however, we only selected methods which does not require any technical expertise from the user. Our final design explains a cluster by using a natural language-based approach to communicate the features that were used to compute the clustering distance function. The intentional shielding of technical information, such as silhouette coefficient, exact feature weights, etc. is to provide a high-level model explanation and not intimidate/overwhelm them with a bag of information that they cannot perceive any way. Our qualitative feedback hints that our approach made Geono-Cluster not only easy to use but also an engaging tool to continue data exploration by rapidly testing different ideas to cluster the data.

Cluster Model Comparison: While viewing multiple clustering results show different ways to partition the data, model comparison to understand trade-offs between these clustering options is critical. However, in our current prototype we do not support explicit cluster model comparison. For example, users cannot perform a pairwise comparison of two cluster models side by side, or they cannot select a few chosen cluster models to see the results in a way which facilitates direct comparison. Based on our interviews with the biologists, comparing cluster models was not posed as a requirement to us. Therefore, we deliberately did not include cluster model comparison as one of the design goals of the system. However, as visual analytics researchers, we understand that being able to compare multiple cluster models, may positively aid model selection and enhance the tools use case.

4.8 Summary

The collaboration with the biologists helped us evaluate our multi-model steering and semi-automatic model selection approaches in real settings. Through this study we clarify assumptions/speculations made when we crafted these technologies. Based on observations and user feedback, we realized there is a wide opportunity for interactive systems that facilitate model construction for non-experts. Further, we noticed a number of other aspects

to the problem of interactive model construction. For example, explaining actions of a model deemed useful for non-experts. Geono-Cluster explains a cluster model by highlighting the top k features that were used to compute the underlying distance function using a natural language expression. Though the simplicity of the explanation was helpful for non-experts, in certain cases this posed a very limited explanation of a clustering model. For example, two cluster models may be based on the same set of features, but the defined clusters may be strikingly different. In addition, annotation of the visual encodings representing model output (the clustered layout) also seemed desirable to the biologists. In the study, some participants expressed their interest in having a feature that enables them to annotate different clusters based on their prior knowledge or insights they gained during data analysis. For instance, one participant mentioned her interest in annotating clusters based on prominent features to help understand cluster compositions easily. We understood that annotation of model outputs is an important functionality that non-experts can use to further communicate their preferences and also to present analytic results to continue their task.

The study also revealed an interesting challenge that many interactive ML systems face, that is - *scalability*. The current interface and the supported interactions (e.g., split and merge technique) is tested with 3000 (approximately) data items. However, we understand that as the size of the data grows, the interaction techniques such as drag-and-drop interaction and lasso-selection tool may be less responsive. This is a concerning challenge for multi-model based VA systems that incorporate custom optimizations derived from user interactions.

In this chapter we described Geono-Cluster that is designed to help biologists visually cluster their data for exploratory analysis. The proposed technique leverage the domain knowledge of the users by allowing a multi-model steering based semi-automatic model selection approach which recommends models according to users' intent. Domain study of these interactive VA techniques of model steering and selection facilitated learning trade-offs of these approaches in a real domain to solve a real problem. Based on collaborative studies with biologists, we built a set of task requirements and design guidelines for our prototype. The technique shown exemplifies a model of interaction which allows non-experts in data science interactively construct clustering models by specifying their preferences. This spares them the burden of going through layers of menus and control panels to transform their expectations to outputs or to comprehend complex model parameters or metrics to find the right clustering model.

CHAPTER V

INTERACTIVE OBJECTIVE FUNCTIONS IN VISUAL ANALYTICS

> **Q3:** *What are the interactive techniques that empower people translate their preferences into objective functions?*

This chapter describes research addressing interactive techniques in VA that enable users to interactively create objective functions to construct classifiers satisfying their preferences.

5.1 Background

5.1.1 Interactive Objective Functions

Machine learning models are driven by the design of objective functions which act as the mathematical expression of preferences, goals, and constraints. ML pipelines must adhere to these considerations when learning from training data to create models. Further, objective functions drive a wide range of ML models serving a specific task, for example, we can input an objective function to any classification model to achieve desired behavior.

However, the specification of such intricate objective functions is difficult for non-experts who are machine learning novices. For instance, creating custom objective functions requires translating one's preferences, goals, and constraints into mathematical expressions by writing code. To support interaction with objective functions or to support interactive construction of objective functions, we need interactive visual interfaces which can help non-experts expressively communicate their preferences or intents to the underlying models. Interactive objective functions can serve as the medium through which people can directly communicate their domain expertise, preferences, and other relevant information to the system. In this chapter, I explain how objective functions can be visualized and made interactive in VA to empower non-experts construct robust ML models.

Types of optimization: Objective functions optimize ML models to conform to user goals. I categorize these model optimizations as external and internal optimization. In the former, an objective function is specified to an Auto-ML model solver, which constructs multiple ML models; then ranks these models by scores computed by the objective function. Next, the system selects top k models with the best performance score. In this approach, models are treated as a black box, meaning internal configuration of the model is not changed or affected; only the model's hyperparameters and learning algorithms are configured to construct new models that may better satisfy user goals. This process follows an iterative search through the high-dimensional model space until a close-fitting model is found (refer Figure 39).

The internal optimization approach optimizes the internal state of a model such as its parameters, weights, and decision making processes to satisfy goals specified in the objective function. For example, in a decision tree model, if the user specifies feature weights or provides a list of important features by order, then we use this knowledge of feature importance to split the decision nodes, instead of using the "criteria" hyperparameter (accepted values are ''gini" or ''entropy") to find features to split nodes by. A similar interactive classification construction using decision trees was shown here [15].

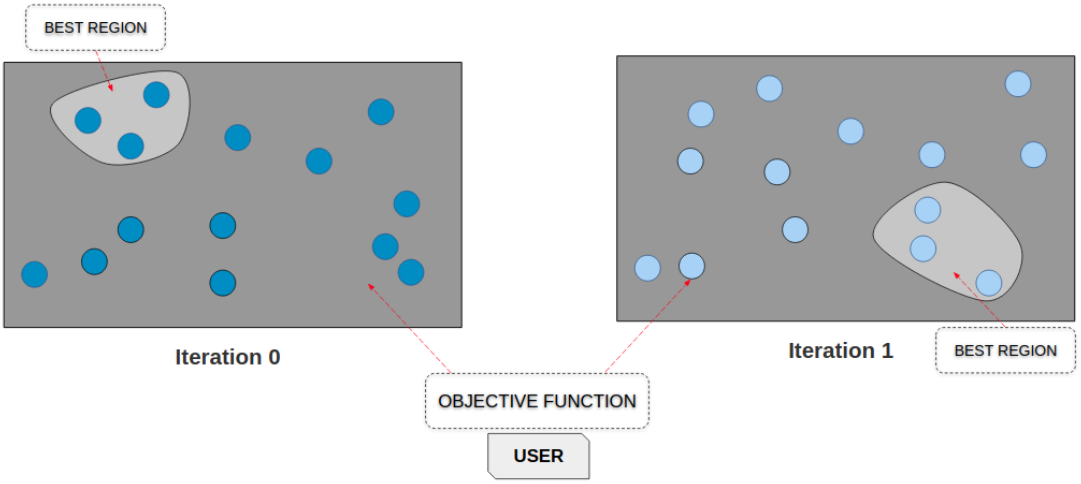


Figure 39: Model sampling from the model space using interactive objective function approach. In this option new models are constructed. New regions of best performing models are found based on specified objective function.

The external optimization technique does not guarantee an optimal model can be found within a specified compute time. However, this technique is model agnostic, meaning that any Auto-ML model solver can take an objective function (specified interactively) to steer/select a set of candidate ML models. On the other hand, internal optimization methods are model specific and may not generalize across different variants of ML models. Thus while an internal optimization approach empowers more flexibility to adjust models, its operations cannot be abstracted to test a diverse set of models using the same VA system.

5.1.2 What is Model Specification?

My research focus resides on user preferences, which are subjective and thus hard to interpret or quantify. In this context I explore techniques that translate user preferences into a set of requirements that a ML model should satisfy. I term this phenomenon as *model specification*. Precisely model specification formalizes user goals/constraints in a mathematical form that models can optimize for. This optimization process may include steering multiple models or utilising an Auto-ML module to select an optimal (preferred) model that satisfies user goals. Model specification accounts for requirements that this optimization process needs to solve for; the optimization process triggers Auto-ML model solver to sample models that are closer to the specifications.

5.2 Challenges in Interactive Objective Functions

While interactive construction of objective functions may sound promising, the implementation of them poses various challenges, some of which are discussed here.

- **R1 - Model tradeoffs and conflicting objectives:** In multi-objective optimization, conflicting objectives and constraints is a major area of concern. In order to make an informed decision, reconciling multiple conflicting objectives is one of the main challenges for users. For example, consider a simpler example from a housing dataset, where a user might want to buy a house within a low budget, with a good

view in the city center. However, the specified preferences conflict with each other (low price vs. good view vs. location). Solving such a query requires matching users' every requirements, which may result in 0 search results. On the other hand the system might report back a set of houses, that satisfies some requirements but not all. These solutions that satisfy some objectives (not all) are called pareto optimal solutions [199]. In various real-world problems users often run multiple model simulations to explore the space of decision choices they have [25]. Using the interactive objective function technique, VA systems can select multiple model options (pareto optimal in nature) as candidate models supporting users' analytical goals. However, a core research challenge lies in that how to explicitly explain such conflicts to users and facilitate model tradeoff analysis through an intuitive visual interface.

- **R2 - Spectrum of objectives and constraints:** Objective functions require objectives and constraints that they should solve. To understand what these different set of objectives and constraints are, we need more research to study how they (objectives and constraints) vary by the given task, data type, or the problem domain. Machine learning models are driven by a wide array of objective functions such as linear, quadratic, polynomial, hinge loss, log loss etc. [195]. Usually, an ML practitioner picks one of the types of objective function based on the data type, the task, and the problem domain. However, when objective functions are made interactive, it becomes challenging to decide on the right objective function type that may best support user goals. Furthermore, user objectives and constraints are not known apriori as they are specified when users interact with VA systems. To understand which objective function type supports which type of user-defined objectives and constraints, and to know what are the various kinds of objectives users can specify for a specific task (e.g., classification, regression, etc.), we need to study the space of objectives and constraints that users can specify (categorised by task and data type).

- **R3 - Design space of interactive objective functions:** Objective functions contain complex mathematical terms (also called sub-objectives) that pose a challenge when we need to visualize them in a form that is understandable by non-experts. The sub-objectives in objective functions have mathematical implications, but when shown in its pure form may overwhelm non-experts. However, to our knowledge, the visual design of objective functions in VA is not explored. Thus further research is needed to understand the design space of interactive objective functions in VA. For example, regularizers such as L2, L1, etc. are hard to describe or visualize to non-experts, but are an integral part of any objective function in machine learning. This requires extensive research to understand how objective functions can be visualized such that it effectively captures user preferences, specifies those preferences to Auto-ML model solvers, and does not overwhelm users. Furthermore, users need to know what are the implications and use-case of specified sub-objectives or constraints in the objective function. If users do not know what are the sub-objectives in an objective function, they may struggle to decipher if the objective function captures there preferences correctly. Here, I think educating non-experts about objective functions, sub-objectives, and other related terminologies (explained in a more straightforward language) may help in creating a better user experience in systems supporting interactive construction of objective functions. For example, to educate users about objective functions,

a VA system can guide users by showing notifications or recommendations of sub-objectives or constraints that they can add, remove, or update. Through my research, I seek to discover various ways ML objective functions can be visually represented and made interactive without overwhelming non-experts.

- **R4 - Evaluation of objective functions:** Correctly capturing user preferences to translate them into an objective function is a difficult task. We need more research to understand how to evaluate the interactive objective functions to probe it's correctness, validity, and fitness to the task, data, and the problem domain. Consider, we design a VA system that helps users interactively construct an objective function. In that system, users can also visually see specified sub-objectives in the objective function. However, how do we know that the objective function correctly captured the user-defined objectives and constraints? How do we ensure that the shape and form of the objective function is of the correct type to minimize the loss in communicating user preferences to the underlying model(s)? There can be various ways to evaluate interactive objective functions with users. For example, controlled lab studies are one viable option where we invite users to interact with a deployed system and ask them to perform a set of pre-defined tasks. Next we ask them questions about their experience and collect qualitative feedback or store log-data, and run quantitative data analysis to seek desired findings, prove the hypothesis to answer specific research questions. However, we need to research extensively to find a logical and scientifically correct method to evaluate interactive objective functions.

- **R5 - Domain study of objective functions:** Studying objective functions through the lens of domain experts who have domain problems and real-world data can validate assumptions, design choices, and rationale behind the implementation of interactive objective functions. But both in HCI and VA, domain studies are difficult for many reasons such as access to domain experts, time, cost-effectiveness, etc. However, if feasible, through domain studies of objective function we may seek answers to questions such as *Does interactive objective functions make sense to domain experts (who are also non-experts in ML)?*, *Do they understand the structural components of an objective function such as sub-objectives, decision variables, constraints etc.?*, *How do domain experts interact with the data to adjust the objective functions to correctly specify their preferences?* We need to evaluate interactive objective functions within specific domains to answer such questions.

Table 6 shows two prototype VA systems that I deployed to test, and address the aforementioned research challenges. This chapter explains QUESTO, while the next explains CACTUS, both of which addresses some of these research challenges. To begin, the next section describes a taxonomy of user-defined objectives for a classification task motivated to address research challenge - **R2**.

5.3 Taxonomy of Constraints

Machine learning algorithms allow for flexible specification of information about data, external knowledge, and context-specific goals and other meta-information through the design of the objective functions that they solve. Our approach of interactive construction of objective functions by users follows a semantic interaction design [72] in that user interactions over data elements in the visualization are translated into objective function terms, that

Table 6: Prototype VA systems in my research that address various research challenges seen in interactive objective functions.

System	Evaluation	Research Challenges	Status
QUESTO	Correctness and effectiveness of Objective Functions	R2, R3, and R4	☑ Completed
CACTUS	Effectiveness of conflict resolution and detection	R1 and R4	☑ Completed

captures user-defined constraints. However, there are numerous constraints or specifications that can be added to objective functions. For example, in a classification task, a user might prefer models which correctly predict specific critical data instance [193], expect to see similar data instances placed in the same class [116, 229], remove data instances which are noise/outliers [228], etc. Based on a literature review of previous work focused on interactive machine learning for non-experts (e.g., [116, 193, 248, 258]), we derived a taxonomy of constraints that can be combined to create objective functions for classification.

Taxonomy Process: Our taxonomy categorizes user preferences based on constraint type to help users build classification models. We studied 61 papers from visual analytics and interactive machine learning area, which included 18 VA systems to formulate the set of constraints under this taxonomy. Using an affinity diagramming approach, we clustered similar papers or VA systems with similar interactions together. For example, based on the literature, we derived constraints such as *Critical* that represents data instances which users find important to be correctly predicted, *Candidate* that captures data instances which are strong representatives of a class label (and not necessarily *Critical*). Further, we iteratively refined these clusters (of paper/VA systems or techniques) until we were satisfied with the relationship between the members of the clusters. We scoped our set of constraints to those which are specified on a models' output relative to the training data. Finally, we derived 15 constraints users can define, organized into 4 categories described below (refer Table 7). To understand the taxonomy, consider the task of a user is to build a supervised model (classification) by demonstrating interactions in any interactive VA system.

1. **Instance-based:** This objective allows users to specify that the classifier should perform well on a set of data instances. While many of the constraints in this section involve adjusting weights on specified data instances, we categorized them separately based on how they are revealed in the user interface and the user task supported.

   -Similarity: This constraint captures the degree of similarity (or difference) between data items. A user specifies a set of similar data instances and expects them to be predicted in the same class label. Users can also specify pairs of data items to be predicted in different classes under this constraint type. Likewise, users may specify a list of pairs of data instances. Each pair represents data items which must be predicted in different classes. However, in both cases ("similar" and "different") users do not specify the class in which the specified data instances should be placed. There are various VA systems/techniques where similarity and difference between data samples

was sought from users to construct models. For example, Tamuz et al. discussed a similarity matrix to infer if an object "A" is similar to "B" or "C" for a user. They asked users: "is object A similar to B or C?" [229] to infer the similarity matrix. Another system Flock asked crowd workers to specify paired examples to define similar or different instances of positive or negative class [45].

-Candidate: This constraint type refers to user feedback on a set of data instances from the training set, which are good representatives of their class. Based on prior knowledge/domain expertize users specify this constraint to inform the system of data instances which represents a given class well. This constraint type was discussed by Zhu et al. [258] in the context of machine teaching.

-Critical: This constraint lets users specify a set of data instances to show that correct prediction of these are critical for them. This constraint can be specified when users want a set of data instances to be correctly classified, while the accuracy of other data instances is less important. For example, consider a financial analysis scenario where a company needs a classifier to predict which clients should be granted a loan. The analyst might know a few clients who are more profitable than others. Thus, he or she might prefer a classifier that correctly predicts these clients than the less profitable ones. Users can assess constructed classifiers based on how accurately they predict the specified critical data instances. Other researchers have looked at critical data instances. For example, Lime is a ML algorithm which helps users to interpret models by explaining their predictions on data instances that are critical [193].

-Ignore: This constraint specifies if the user intends to remove noisy or outlier data instances from the training set. Removal of noise or an outlier increases the accuracy of prediction and the power of the model to generalize on unseen data items. This constraint may also include specification of data instances whose prediction (correct or incorrect) is irrelevant to users. Elzen et al. prototyped a system BaobabView that enables users to inspect outliers and noisy data samples through dot plots to improve interactive construction of decision trees [236].

Table 7: Taxonomy of constraints defining categories and sub-categories. For each sub-category, it lists relevant works in VA literature that show similar constraint specification.

Category	Sub-Category	Relevant Works
Instance-based	**Similarity**	**[44, 45, 56, 119, 121, 229]**
	Candidate	**[44, 121, 240, 258]**
	Critical	**[50, 116, 193, 204]**
	Ignore	**[56, 119, 236]**
Feature-based	**Feature Selection**	**[15, 45, 129, 147],**
	Correlation and Variance	**[15, 94],**
Train-objectives	**Accuracy, Precision,**	**[129, 193, 228, 244, 255]**
	Recall, and F1-Score	**[42, 79]**
Test-objectives	**Test-Accuracy, Test-Precision,** **Test-Recall, and Test-F1-Score**	**[14, 79, 181, 201, 214, 243]**

2. **Feature-based:** This category includes items which users can specify to help a model

in its learning process. The following constraints defined under this category operates at the level of the features (or attributes).

-Feature selection: This constraint allows users to specify features that are important for classification. A classifier will only use the specified features when it is learning the data. This constraint is derived from the previous work in the literature showing the value of human-centered feature selection in model construction [45, 129, 147].

-Correlation and Variance: This constraint allows users to specify correlation and variance in the features, based on their domain expertise. Users specify perceived correlation between features and variance per feature in the data. *Correlation* and *Variance* are based on the user's domain knowledge and not necessarily grounded in the training data. For example, a financial analyst might know that experienced bank customers in the age group of $40-50$ are more likely to spend economically and pay on time. Thus the features *age* and *payment* are correlated based on the domain knowledge of the user, which may or may not be present in the data. In the literature, Hall showed a feature selection method based on correlation in the data[94]. Instead of computing correlation in the features from the training data, this constraint enables users to specify correlation and variance in the data, an information that affects how the model learns from the data.

3. **Train-objectives:**

   ML models can be evaluated using conventional model metrics or constraints such as **Accuracy**, **Precision**, **Recall**, and **F1-Score**. When applied to the training set, we consider them *Train-objectives*.

4. **Test-objectives:** When the aforemention metrics (**Accuracy**, **Precision**, **Recall**, and **F1-Score**) are applied to the test or validation set, we consider them *Test-objectives*. Together these two objectives (Train and Test objectives) help users control for model overfitting. Using the *Train-objectives* users can verify how well the model is learning from the data, while using the *Test-objectives* they can validate if the model generalises well on unseen samples. For example, one can first find models with high training accuracy (i.e., low bias), then tune the hyperparameters to achieve a high test set accuracy (i.e., low variance), and finally weight or rank these constraints to find classifiers that show an optimal tradeoff between bias and variance.

Discussion: It is difficult to construct a taxonomy that perfectly characterizes a domain or every ML problem/task. To our knowledge, in the VA community there is no taxonomy that categorizes user-defined constraints specified to interactively construct classification models. Our exercise while captures how the other VA researchers have utilised user-defined constraints, has a set of drawbacks. For example, constructing a taxonomy for such a task that may also be further categorized based on user expertise in ML (non-experts, intermediates, to expert ML practitioners). From the extensive literature review and prior experience in the field, we formalized the taxonomy of constraints, though the categories in it are not exhaustive. We believe this is one of the approaches to categorize the user-defined constraints to understand how to build VA systems and interactions that can capture user preferences to hep users construct and select classifiers. That being said, we are aware that more research is needed to further refine (add, remove or update) these categories as new VA systems/techniques are formulated.

This taxonomy of user-defined constraints (for classification tasks) helped us understand and address the research challenge **R2**. While the taxonomy was for a specific ML task, moving forward, we need similar taxonomies for other ML tasks such as clustering, regression, graph matching, etc. However, we can expect that many of these task-specific taxonomies may show overlapping constraints. For example, the constraint type *Similarity* is relevant and useful for a clustering task too. Furthermore, building on this taxonomy, I designed a VA system QUESTO specifically supporting classification task for power users using an interactive objective function approach.

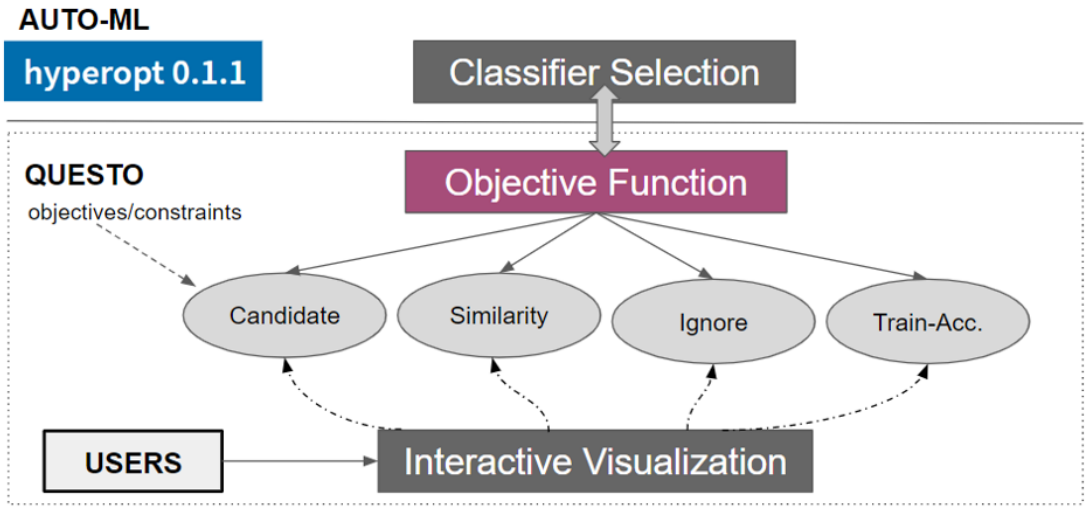


Figure 40: The working logic of QUESTO coupled with an Auto-ML optimizer - Hyperopt. While some user interactions directly specify constraints to the objective functions, others are automatically inferred by QUESTO.

5.4 QUESTO - an interactive construction of objective functions

We hypothesize that using an interactive objective function users are empowered to specify desired goals without the need to directly interact with the complex mathematical terms. Here we describe a prototype VA system QUESTO that translates user interactions with data into objective functions for a classification task. Specifically, QUESTO allows users to specify function terms (also called sub-objectives[202]) in the context of multi-objective optimization problems. QUESTO facilitates the construction of a classifier by allowing users to formulate their preferences as an objective function while they explore and interact with a tabular dataset (see Figure 40). Furthermore, the system visualizes the underlying objective function to help users review specified goals and constraints.

5.4.1 UI - Main Views

The UI of QUESTO has four main views - (1) Data Table, (2) Scatterplot Matrix, (3) Confusion Matrices, and (4) Feature Panel.

Data Table View: This view supports 4 types of constraints: *Critical*, *Ignore*, *Similarity*, and *Candidate*. It displays the training and test sets in two data tables. To specify *Critical* or *Ignore* constraint, users can click on the row (See Figure 44-B). Further, users can assign weights to the specified *Critical* data instances, up arrow means a high weight, down arrow means a low weight). To specify a *Similarity* constraint, users can select the constraint from

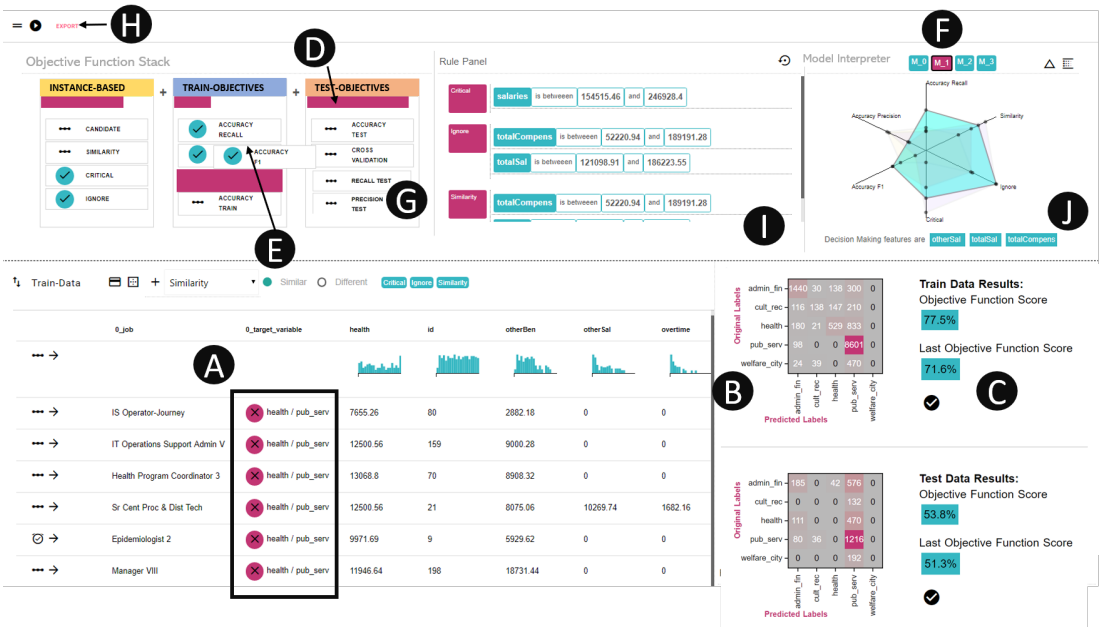


Figure 41: QUESTO: A. Incorrect predictions. B. Confusion matrices. C. Model score. D. Sliders to weight objectives. E. Draggable constraints to specify priority. F. Selected model. G. Test data constraints. H. Model run and Export button. I. Rule panel. J. Important features.

a drop-down menu and then specify either "Similar" or "Different" on pairs of data instances (by clicking on the row). Similarly, to specify the *Candidate* constraint (see Figure 45-A), users can select rows within a class to specify example data instances. For both *Similarity* and *Candidate* constraints, the view highlights the selected rows with a green color (see Figure 44-B).

Scatterplot Matrix View: The scatterplot matrix shows relationship between various data attributes such as correlation, variance, etc. (see Figure 45-B). Further, it helps users to find noise, missing data, outliers, etc. Users can select data instances by brushing to specify examples for constraints such as *Critical*, *Ignore*, *Candidate*, etc.

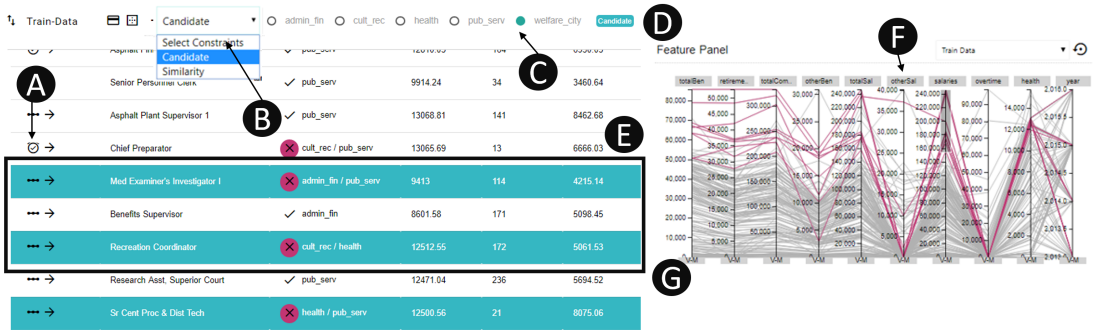


Figure 42: A. Critical constraint. B. Other constraints. C. Class selector. D. Filter tags. E. Selected data. F. Select features, correlation, and variance. G. Parallel coordinate plot.

Confusion Matrix: The confusion matrix for both the training and test set is shown on the right of the data table view (see Figure 41-G). Hovering or clicking on any of the cells in the confusion matrix filters the data table to show data instances responsible for the

predictions in the cell. Next to the confusion matrix is a description of the model's (current and previous) objective function score (e.g., "Objective Function Score is 76%"). Users can hover over the previous score to compare it's confusion matrix with the current model.
Feature Panel: The attributes of the data are visualized as a parallel coordinate plot (see Figure 41-F). Brushing on the features (represented as vertical lines) allow users to filter the data instances shown in the data table or the scatterplot matrix view. Users can specify a *Feature Selection* constraint by clicking on the top headers over the vertical lines (see Figure 41-F). In addition, right-clicking on the gray box, users can specify a *Correlation and Variance* constraint.

5.4.2 UI - On Demand Views

The following views are activated by users on demand. They are seen by clicking on the top bar in QUESTO's interface.
Objective Function Stack: This view visualizes the interactive objective function by showing the set of constraints users can specify. Each box (see Figure 41-A) represents constraints under the categorization described in our taxonomy. Users can specify a constraint by clicking on its name. A specified constraint is shown with a green checkmark (see Figure 41-A). Based on how users explore the data in the main views, QUESTO automatically sets constraints in this view. Users can drag sliders to adjust the weight of each constraint category individually. They can also drag-drop constraints within each category to specify relative ordering.
Model Interpreter: QUESTO visualizes k ML models and their metrics in this view. Users can choose to inspect each model by seeing it in a parallel coordinate view or in a star plot view (see Figure 41-E). Users can select a model, which updates the prediction column in the data table view and the adjoining confusion matrix view.
Rule Panel: When users specify any constraint to the objective function based on a set of filtered data instances, QUESTO saves these data instances as part of a *rule* (See Figure 41-H). Users can assign custom names to them. The rule highlights the feature values based on which the filtered data instances were added to the specified constraints. Hovering over them, users can see the set of data instances under the rule in the constraints list view.
Constraints List View: This view shows user-specified constraints as a list, where each row represents a data instance. Further, it shows constraints that are satisfied by the selected model with a checkmark.

5.4.3 Technique

In this section we describe the underlying techniques in QUESTO that drive the construction of an interactive objective function and use it to create classification model (see Figure 43).

QUESTO uses an Auto-ML module called Hyperopt [127] to find the optimal hyperparameter combination to construct classifiers aligned with user goals. We used a Random Forest classifier with the hyperparameters *MaxDepth*, *Criteria* ("gini" or "entropy"), and *MinSamples*. Users formulate their preferences in the objective function O by specifying a set of constraints Φ. Using the interactions in QUESTO, users define a weight vector W. In each iteration, Hyperopt consumes O to construct and rank new models M (based on their score S). This iterative cycle allows users to search for models by defining their goals and constraints in O while inspecting the model performance.

5.4.3.1 Classifier Creation and Selection

Model Construction: QUESTO splits the loaded data set C, into training D and test set T. QUESTO triggers Hyperopt to start the model optimization process (explained below) on the supplied data D and T to build M classifiers of size N ($M = m_1, m_2, m_3, ...m_N$). It constructs each model m_i using a supplied learning algorithm ω and sampled B hyper-parameter combination such as $\lambda = \lambda_1, \lambda_2, \lambda_B$. The models are evaluated based on the accuracy of their predictions on the training data, and the current objective function.

Model Optimization: Hyperopt traverses the space of pre-defined hyperparameters to construct a set of models. As an input, Hyperopt expects a list of hyperparameters to tune, $H = h_1, h_2...h_B$ and a domain space ν_i for each hyperparameter h_i. We pre-selected B hyperparameters and their domain space. Using Hyperopt's hyperparameter tuning, our technique constructs N ML models (M) and evaluates each of them using a supplied objective function O. The objective function O is constructed by a weighted linear combination of user-defined constraints or sub-objectives Φ. Hyperopt ranks the N classifiers (we set $N = 300$) based on the objective function scores in S.

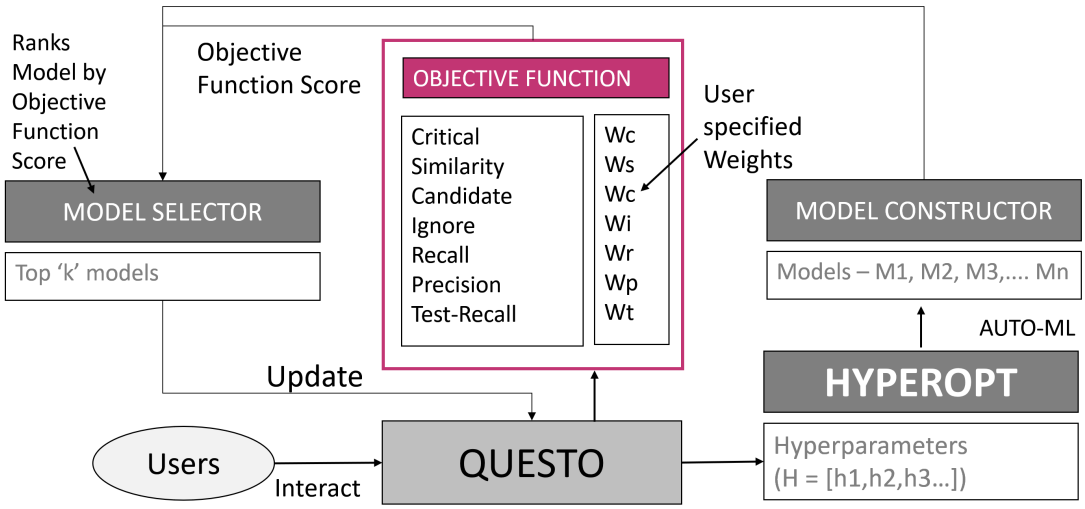


Figure 43: QUESTO system architecture.

5.4.3.2 Interactive Objective Function Creation

Our technique translates user interactions into a set of sub-objectives (or constraints) Φ and assigns scores to them (S) to compose the multi-objective objective function O. In QUESTO, users can define the following sub-objectives interactively:

Critical: Hyperopt trains a model m_i on the supplied training set D with original labels $L_{d,o}$. Next it predicts labels on the training set as $L_{d,p}$. Users specify a list of IDs of x critical data instances $C = c_1, c_2..c_x$. QUESTO retrieves the prediction on the critical data instances based on supplied IDs C as $L_{c,p}$ (Note $L_{c,p} \subset L_{d,p}$). Similarly, QUESTO retrieves the original class labels of the critical data instances as $L_{c,o}$. QUESTO initializes the score

for the *Critical* constraint as $s_1 = 0$. We compare $L_{c,o}$ with $L_{c,p}$, to find the number of correct matches and save in s_1. We normalize s_1 to get a score between 0 to 1.

Similarity: A user specifies a list of data items V that is similar. The algorithm iterates over V to find the original class label $L_{n,o}$. If V belongs to more than one class label, the most frequent class label is assigned to $L_{n,o}$. Next, the algorithm matches the prediction $L_{n,p}$ with the original class label $L_{n,o}$, where i is the index, iterating from 0 to b (the number of similar data instances specified by the user). For every correct match, the score for this sub-objective $s_{2,a}$ gets a +1 point. Next, it normalizes $s_{2,a}$ as $(s_{2,a})/b$, to get a score between 0 to 1.

A user specifies a list of tuples σ, where each item in the list are expected to be predicted in different classes. It is represented as $\sigma_i = (\sigma_x, \sigma_y)$. The algorithm iterates over σ and matches the predicted class label $L_{\sigma,p,i}$ of item σ_i to class label $L_{\sigma,p,j}$ of item σ_j. For every incorrect match, the score for this sub-objective $s_{2,b}$ gets a +1 point. Next, it normalizes $s_{2,b}$ as $(s_{2,b})/b$, to get a score between 0 to 1. Finally the score for this sub-objective is computed as $s_2 = (s_{2,a} + s_{2,b})$. We normalize s_2 to get a score between 0 to 1.

Candidate: Users specify a list of b data items G for a class label A. The algorithm increases the training data weights ($W = w_1, w_{s2}, w_{s3}...W_b$) for these data items. It inputs W to Hyperopt which trains the classifiers M based on the updated weights. The algorithm iterates through G and matches each items predicted class label $L_{f,p}$ with A. The score for this metric s_3 is initialized as 0. For each correct match in the iteration, s_3 is assigned a +1 point. Finally the algorithm normalizes s_3 as $s_3 = s_3/b$.

Ignore: Users specify a list of data items I. The algorithms remove these data items from the training set D to form a new training set D_{II}. Further when computing classification model metrics such as *Precision*, *Recall*, etc. the algorithm removes the data items present in I from D and T.

Accuracies: This captures the classification model metrics defined as part of the category *Quantitative* and *Generalization* (see section 5.3). These are *Precision*, *Recall*, *F1-Score*, etc.

Feature Selection: Users can guide feature selection in one of two ways. First, users can select a set of features F_d using the filter panel. QUESTO interprets that the user chose a set of features which has a high correlation with the class label, and thus will only use these features F_d. Second, users can specify a set of correlated features F_c. QUESTO automatically discards these features as they show a high correlation (negative or positive) with each other.

Objective Function Formulation: The resulting objective function in QUESTO is a weighted linear combination of sub-objectives Φ. The algorithm computes the scores for each constraint. Finally, it linearly combines these individual sub-objectives scores as shown in the equation below:

$$O = s_1 * \omega_1 + s_2 * \omega_2 + s_3 * \omega_3 + s_4 * \omega_4 + s_5 * \omega_5 \quad (1)$$

where s_1, s_2, s_3 are *Critical*, *Similarity* and *Candidate* sub-objective scores respectively, and $s_4 and s_5$, represents *Precision* and *F1-Score* sub-objective scores respectively.

Weighting Preferences: Initially the weights $W = \omega_1, \omega_2, \omega_3 .... \omega_U$ (sums to 1) are evenly set for each sub-objectives in Φ However, users can override and specify weightings for each sub-objective by aforementioned interaction techniques.

Figure 44: QUESTO loaded with the diabetes dataset - A. Rule panel showing saved rule. B. Selected data row (shown in green) given as an example for *Candidate* constraint. C. Incorrect prediction of a data item. D. User can brush over Feature panel to create a rule and also can filter data items. E. Model interpreter showing a list of available models and the star plot view. F. A drop-down selector to pick additional constraints.

5.4.4 Usage Scenario

Lets' understand how QUESTO can help non-experts interactively construct an objective function to build a classifier. Shane is a medical practitioner and has a dataset of 1000 diabetic patient records from 1999-2008 [111]. The data contains 20 attributes such as *race*, *gender*, *insulin level*, *number of times inpatient*, *time in hospital*, *medical speciality*, etc. The target label in the data predicts if a patient will be readmitted as a diabetic patient. The labels are "greater than 30%", "less than 30%", and "NO".

Shane imports the data in QUESTO and presses the "build model" button (see Figure 44 top left), which results in QUESTO calling Hyperopt to build 200 classifiers and visualize the top 4 in the model interpreter view. Shane selects model 4 to check the confusion matrices on the right of the data table (See Figure 45-E); he sees the current model performs poorly (avg. 53 % accuracy) on both the training and test sets. Shane filters the male patients who are *caucasian*, and have high values in the features -*insulin level* and *time in hospital* by brushing on the feature panel (see Figure 44-D). He sees that they are incorrectly predicted (originally labeled as *greater than 30%*). He selects the *Candidate* metric from the drop-down menu shown in Figure 44-F and specifies example patients to the label *greater than 30%*. QUESTO saves Shane's specifications as a *rule* in the rule panel (see Figure 44-A), based on the feature values in *race*, *insulin level*, and *time in hospital*. He renames the rule as *caucasian-high-risk*.

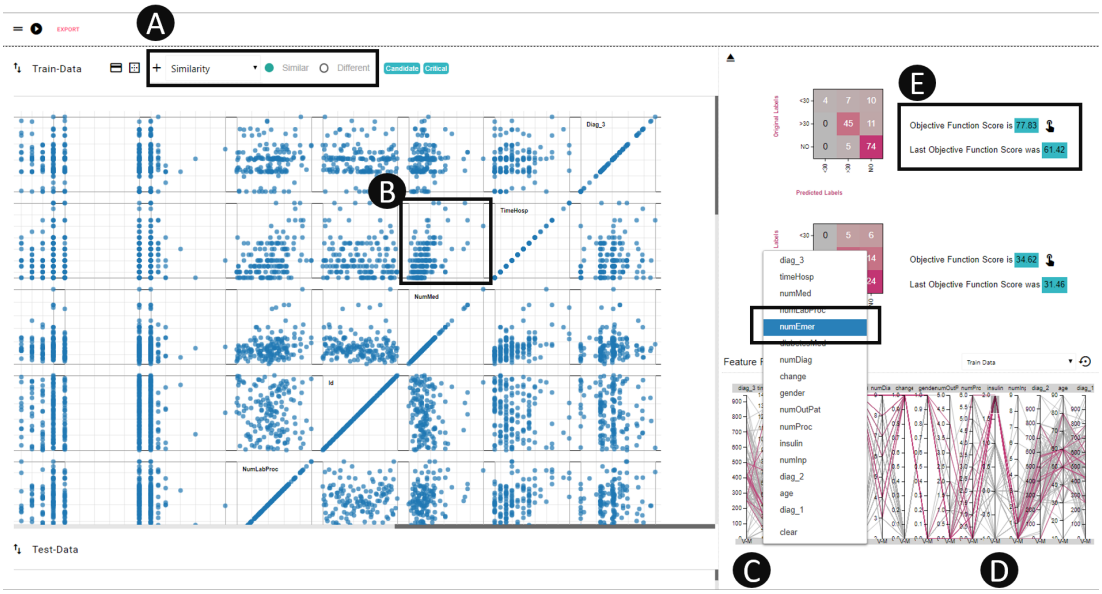


Figure 45: QUESTO loaded with the diabetes data - A. The *Similarity* metric from the drop-down selector. B. The scatterplot matrix view showing relationship between various attributes. C. User right clicks on the Feature panel to specify correlation between the feature. D. User can click on the boxes to specify features with high/low variance and correlation. E. Shows the model performance improved on the training data.

Shane constructs new classifiers (see Figure 44-E) and notices in the model interpreter view that the best model from the set improved the objective function score to 61%. He hovers on the rule *caucasian-high-risk* to see that approximately 75% of the patients in the rule are correctly predicted. Shane clicks on the drop-down menu from the data table view (See Figure 45-A) to select the *Similarity* constraint. Next, he clicks on the rows of the training set in the data table to specify a subset of these patients (that are incorrectly predicted) as examples for the *Similarity* constraint to specifying similarity among these patients. Based on observations from scatterplot matrix (See Figure 45-B), Shane specifies correlation among the features - *time in hospital* and *number of medication* in the feature panel (See Figure 45-C).

Shane builds new models and hovers over the star plot view (see Figure 44-E) to find model 1 performs best on the specified constraints. He hovers on the rule *caucasian-high-risk* to see that all the patients in the rule are correctly predicted on the training set by this model. However, on the test set Shane sees a low accuracy of 70%. In the objective function stack, Shane removes the *F1-Score* constraint and adds *Testing Accuracy* and *Cross-val-score* constraint to build new models.

Shane inspects the new classifiers and finds that the patients in the *caucasian-high-risk* rule are correctly predicted and the performance on the test set improved as well (to 83%). Feeling confident Shane exports the selected model to predict new patients in the future. This usage scenario showed how a domain expert user could fulfill very personalized expectations from a model by interactively specifying goals and constraints to an objective function.

5.5 QUESTO - Evaluation

We ran a pilot study of QUESTO with 6 users to get early feedback on the system and its usability. Through the pilot study we qualitatively observed user responses to interactive objective functions. On a high level, participants found the workflow presented in QUESTO easy to use, intuitive, and engaging. They found interactive objective functions, an appropriate approach to adjust models without getting into the mathematics and theoretical aspects of machine learning. However in some cases, they felt uncertain and confused about objectives that they need to specify, e.g, few participants asked: *What should I do next to improve the classifier's accuracy?* or *What did I do wrong that the classifier's accuracy degraded?*

Building on the feedback from the pilot study, we conducted a within-subject user study of QUESTO comparing it with approaches to interactively build classifiers. For example, other than using interactive visual interfaces like QUESTO, two popular alternatives to constructing classifiers are: (1) manually creating them via code or command-line (CMD) interfaces, and (2) automatically generating them using Auto-ML. We wanted to compare QUESTO with manually coding to verify if QUESTO is easier and faster to specify constraints. Furthermore, we wanted to compare QUESTO with Auto-ML to verify if QUESTO satisfies subjective user goals better, and to compare the resulting accuracy. We conducted a within-subject controlled lab user study of QUESTO comparing it with CMD, and Auto-ML to construct a classifier. Our study addresses the following research questions:

RQ1 Is QUESTO easier and faster to use than CMD workflows?

RQ2 Does QUESTO build more accurate models than automatically-generated models from Auto-ML?

RQ3 Does QUESTO build models that addresses user-specified constraints better than models from Auto-ML?

We hypothesize the interactive visual interface of QUESTO will be easier and faster to use compared to command-line interfaces. Here "faster" refers to the time it takes to specify constraints and construct classifiers; it does not include the time needed to train classifiers. Further, we hypothesize that Auto-ML will generate more globally accurate models, given the objective function optimizing towards a specified accuracy metric such as high cross-validation score. However, **RQ3** tests our hypothesis that QUESTO will be better at building models that fit specific user constraints given the customized objective function, thus illuminating this tradeoff between accuracy and user goals.

In the literature there is no well-defined metric to measure how well user-defined constraints are satisfied in classifier construction. Thus, we defined **constraint satisfaction score** as a metric that captures how well a model attains user-specifed constraints such as *Critical*, *Candidate*, etc. These scores are normalized between $0-1$ (higher is better). It is expressed as $(\sum_0^U \omega_i * \gamma_i)/\mathbb{U}$, where ω is the weight of U constraints, γ is the score of each constraint, $\mathbb{U}$ is the total number of specified constraints in the objective function.

We recruited 16 participants (7 Male, 9 Female), aged between 21-29 (M=25, SD=2.91), by inviting participants through our university mailing lists. We required them to at least have a basic expertise in writing python code with elementary understanding of ML. All of the participants (undergraduate and graduate students) were familiar with basic/intermediate data analysis using tools such as MS Excel, Tableau, etc. The experiment lasted 60 minutes and we compensated each participants with a $10 Amazon gift card. The study was conducted using a 17-inch display and a mouse.

5.5.1 Study Design

We began the study with a practice session teaching users how to interact with both QUESTO and CMD (with VS Code as the scripting editor). During the practice session, participants were encouraged to ask questions to clarify uncertainty in relation to the workflow or the interaction design. We proceeded to the experimental sessions only when the participants felt confident in using each system. For quantitative evaluation each participant interacted with both QUESTO and CMD using one dataset. Furthermore, the order of the interfaces (QUESTO and CMD), the datasets, and the tasks were randomized to remove any ordering/learning effect. We considered these dependent variables for quantitative analysis: (1) *Task completition times:* in specifying an objective function through QUESTO or CMD, (2) *Model accuracy:* the accuracy of the model constructed using QUESTO, CMD, and Auto-ML stand alone, (3) *Constraint satisfaction score:* measure to capture constraints satisfied by the model, and (4) *User preference rating:* average preference rating of *Ease of Use* for QUESTO and CMD.

5.5.2 Datasets

For the practice session, we provided a dataset of 10000 credit card transaction records [250]. The data has attributes such as *bill paid month 1*, *bill paid month 2*, *bill paid month 3*, *account balance*, etc. It was a binary classification task to predict if a bank customer will default on a bank loan or not. For the experimental sessions, we provided San Francisco's employment dataset [165] containing 5000 data items for the quantitative evaluation. Each row in the data contains information about an employee's remuneration containing attributes such as *dental benefits*, *annual salary*, *monthly salary*, *extra benefits*, etc. The task was to predict the employee's department (5 classes). For the qualitative evaluation we used a movies dataset [3] (5000 samples) containing attributes such as *budget*, *gross revenue*, *facebook likes of lead actors, director, cast*, etc. The data has three labels for movie rating: high, medium, and low. Following best ML practices, we use multiple test datasets, so that the constructed model never sees unseen data instances,

5.5.3 Tasks and Procedure

In the practice session, we asked users to perform 2 tasks per interface. The first task was to design an objective function in QUESTO by specifying a *Critical* constraint. They were asked to iterate multiple times and improve the classifiers performance by refining the constraint. The next task was to add two more constraints (*Similarity*, and *Candidate*) to the objective function in QUESTO and iteratively improve the classifiers performance. They repeat the same tasks by writing code and using CMD to run their code to construct a classifier. For the quantitative evaluation participants are randomly assigned an interface (QUESTO or CMD). Participants were encouraged to think aloud while they interact with each system. The interviewer was a silent observer and was away from the participant to mitigate *Hawthorne and Rosenthal* effect during the session. Participants performed 3 tasks per interface (6 tasks total). Each of the tasks were in increasing level of difficulty.

Task 1 Specify the constraint *Similarity*. (Max time: 3 minutes)

Task 2 Specify *Critical*, and *Candidate*. (Max time: 5 minutes)

Task 3 Specify *Critical*, *Similarity*, *Candidate*, *F1-Score*, *Precision*, *Accuracy*, and *Cross-Validation*. (Max time: 7 minutes)

Figure 46: The study results highlights differences in constraints satisfactore score and model accuracy, comparing QUESTO with coding interfaces and Auto-ML techniques for classification tasks.

For the qualitative evaluation, we asked participants to freely use QUESTO (may specify any constraint) to build a classifier (on the movies data), and improve its objective function score in 5 minutes.

5.5.4 Data Collection

For quantitative assessment, we saved log data which stores (per iteration) models selected by users, their learning algorithms, and hyperparameters, predicted class labels, interacted data items, user-specified constraints etc. When participants completed the three tasks on both interfaces, we asked them to fill a NASA-TLX form [93], and a post-study questionnaire with a set of likert scale questions. We asked questions to seek user preference rating to each interface's various dimensions such as: (1) Ease of use, (2) Flexibility, (3) Fun to use, (4) Learnability, and (5) Intuitiveness. After the qualitative evaluation we conducted a semi-structured interview asking open-ended questions about the workflow, system usability, and interaction design for each interface. In this interview we asked questions such as: (1) *Describe your thought process while you interacted with QUESTO?*, (2) *Explain your experience in constructing an objective function interactively vs, through writing code?*, (3) *How do you think we can improve the current workflow, and design of QUESTO?* We also captured video and audio of participants screen while they interacted with QUESTO.

5.5.5 Quantitative Analysis

For the following we used the Friedman Test for Repeated-Measures as it is a good indicator of statistical significance for multi-class classifiers with multiple datasets, which may not follow a normal distribution as suggested by [59]. Furthermore, we utilised Post-hoc Wilcoxon signed-rank tests with Bonferroni correction (new p value = 0.03) to test for statistical significance.

Ease of use: To answer **RQ1** we used likert scale rating scores (5-point scale) from participants (average ease of use rating 4.64, higher is better, see Figure **??**-D). The ease of

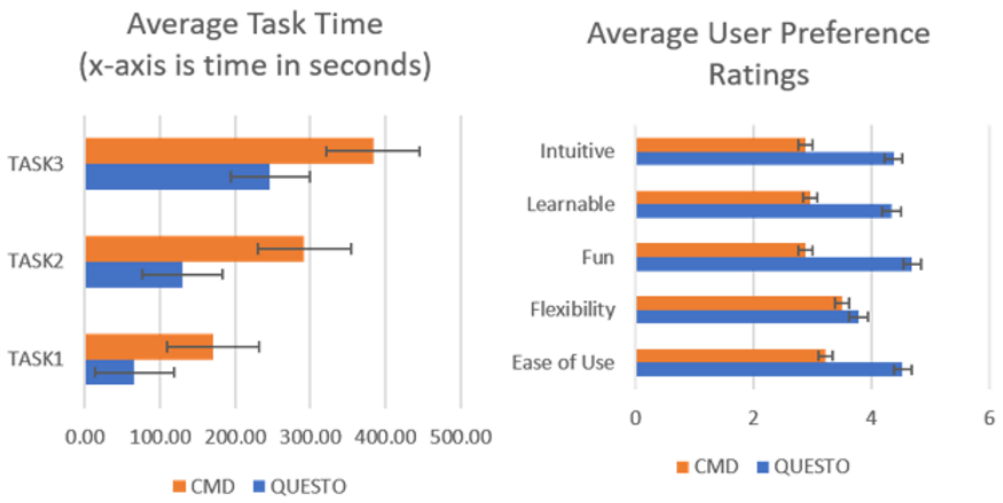


Figure 47: The study results showing average task completition time and average user preference ratings, comparing QUESTO with coding interfaces and Auto-ML techniques for classification tasks.

use of QUESTO was significant across all tasks: $(\chi^2(1) = 48.03, p < 0.03)$. thus answering **RQ1** that QUESTO is easier to use than CMD.

Task completion times: Participants were significantly faster to specify constraints in QUESTO than CMD (Figure 47). Every participant completed all tasks except for three who failed to complete task 3 in the allotted time using CMD. In these cases, we recorded the maximum alotted time for that task. Quantitatively we found statistically significant difference in task completion time for all tasks: Task1 $(\chi^2(1) = 16.0, p < 0.03)$, Task2 $(\chi^2(1) = 16.0, p < 0.03)$, and Task3 $(\chi^2(1) = 4.0, p < 0.03)$. This answered **RQ1** that QUESTO is faster to specify constraints than CMD.

Model Accuracies: To answer **RQ2**, we compared the models generated using QUESTO, CMD, and Auto-ML with respect to their accuracies. We found QUESTO generated similar quality models in relation to accuracies as produced by CMD (refer Figure **??**-A). However, accuracies generated by models from Auto-ML stand alone were higher in comparison to QUESTO. We observed statistically significant difference in the model accuracy for task 1 $(\chi^2(1) = 12.25, p < 0.03)$, and task 3 $(\chi^2(1) = 4.0, p < 0.03)$.

Constraint satisfaction: We found that QUESTO outperformed Auto-ML in meeting user-specified constraints for all tasks: Task1 $(\chi^2(1) = 6.25, p < 0.03)$, Task2 $(\chi^2(1) = 2.25, p < 0.03)$, and Task3 $(\chi^2(1) = 9.0, p < 0.03)$. This answered **RQ3**. We analysed this based on data collected from the employment dataset [165] (users were given the example data items to specify as constraints), and on the movie dataset [3] (users freely specified constraints, see Figure 46).

5.5.6 Qualitative Analysis

User Preferences: We measured user preferences using 5-point likert scale rating. QUESTO's user preference ratings (out of 5) were higher than CMD workflow in various aspects such as, *Easy of use* (Q: 4.53, CMD: 3.22), *Fun to use* (Q: 4.69, CMD: 2.88) , *Learnable* (Q: 4.34, CMD: 2.97), and *Intuitive* (Q: 4.38, CMD: 2.87) (see Figure 47). In addition, through

the NASA-TLX survey we found that average participants' cognitive workload was significantly lower in QUESTO than CMD interfaces (Q: 4.30, CMD: 8.87, out of 10; lower is better). Based on these results it is likely that participants preferred QUESTO compared to using the CMD interface for objective function creation. However, many participants found CMD comparably flexible to QUESTO to specify preferences (Q: 3.78, CMD: 3.52) and debug models.

Easy workflow: Every participant found QUESTO's interface and the workflow easy to learn and use. Users liked the table view, with the ability to filter and search for specific data items as examples for constraint specification. P03 said, *"I like how I can mouseover on the cells of the confusion matrix to see incorrectly predicted data items."* P08 added, *"I frequently brush over the feature panel to filter data items by a set of attributes I care about."* However, users recommended that more advanced sorting feature in the table would have helped them find appropriate examples quickly.

Level of Detail: Some participants had elementary knowledge of python and ML, while a few others had in-depth ML expertise. P09 reflected *"Best part of this workflow is how easy the tool makes to change my constraints in terms of weighting or ordering them, which lets me look into a new set of classifiers"*. While advanced ML users appreciated the idea of an interactive interface to define an objective function, reflected that they would prefer to look at the numerical weights on each constraint so that they can better trust how the modeling engine is selecting classifiers. P13 said *"Though the objective function view is very useful, I am not sure if I understand what's happening on the background without knowing the numerical weights on these constraints.*

Meaningful constraints: Participants found the constraints in QUESTO appropriate and useful for classifier construction. P10 added *"I think the constraints make a lot of sense to me. When I test models I frequently look at specific data items to verify if the classifier modeled the data correctly"*. However, a few participants commented that it would help if QUESTO could recommend items to consider next. For example, P03 suggested *"I think you can use prediction probabilities to reflect on how confident the model is on each prediction. That may help me specify better examples."*

Intuitive model selection: Users found the model interpreter view useful to compare models based on their performance on the specified constraints. P17 noted *"From the star plot view, I can inspect the size of the polygons to select models that performed better on relevant constraints. I would prefer to mouseover (instead of click) to browse the model output results on the table and the confusion matrices."* However, participants wanted to see a bookmark feature to save models they like. Furthermore, one participant desired to see how each constraint contributed to a change in model output to understand what interaction improved performance.

Task complexity: The three tasks in the user study had different levels of complexity. While task 1 and task 2 involved satisfying only subjective user preferences, task 3 included finding models that are accurate yet address constraints such as *Similarity*, *Candidate*, etc. Participants found task 3 more challenging, as finding the right model that is accurate and addresses their personal goals was difficult. P15 noted *"It was hard to improve the classifiers performance when I had more than 4 constraints to specify. However, with QUESTO I could rapidly test different weightings/rankings to these constraints to find an optimal classifier"*.

Conflicting constraints: We observed that participants sometimes specified conflicting constraints. For example, they specified a set of data item as a *Candidate* constraint, but in a later iteration they specified a subset of these data items as a *Ignore* constraint. As QUESTO currently does not support alerting users about conflicts, in future we plan to

mitigate them in objective specification.

Modeling process: From the study we found that QUESTO's workflow does not help users understand the underlying modeling process. Often participants wondered what interaction in the previous iteration led to the improvement or degradation in the accuracy or the objective function score. Future work could make the modeling process more transparent and allow users to reason about the impact of their objective specification on classifier selection.

5.6 Discussion

User-System Feedback loop: Interactive objective functions may guide an Auto-ML model solver to select models that weigh data instances preferentially to better support user goals. For example, an objective function with the *Critical* constraint will guide the Auto-ML solver to prefer classifiers that correctly predict specific items. In response, users may inspect a model in relation to how well it supports the specified constraints. Thus interactive objective functions employ a two-way feedback loop between the user and the system for: (1) communicating preferences to find suitable models, and (2) providing a medium for understanding classifier performance. Participants confirmed that (per iteration) they validated models in relation to how well they support the constraints.

Tradeoff Analysis: Practically, in a multi-objective optimization problem, satisfying every objective might not be feasible [202] due to a plethora of reasons including conflicting constraints, mathematical infeasibility, noise/outliers in the data, high dimensionality, etc. [116]. Formalizing user goals as a set of constraints in an objective function facilitates searching for a set of pareto-optimal solutions which may only satisfy a subset of the specified user goals. Furthermore, visualizing these pareto-optimal models help users inspect them and understand tradeoffs.

ML users often seek models that support other customized subjective goals/objectives that are personal and problem-specific [116]. This study validated that while QUESTO produced slightly less accurate models than Auto-ML model solvers, they met subjective user-specified constraints. We envision that users will need to find a balance between these two extremes.

Scope and assumptions of the user study: In the study we timed each task to ensure the study can be completed within a reasonable time. However, in more realistic settings, optimizing for (or limiting) time may not be meaningful. Further, we used pre-defined hyperameters within which Hyperopt sampled values to construct classifiers. The study results may vary if the set of hyperparameters used are different. Also, our study only included participants with basic expertise of python and ML.

Model overfitting: Good ML practices entail that trained models have no knowledge of test data. In QUESTO we follow the same process. The test data view in the data table allows users to inspect classifiers by reviewing the predictions made at the data instance level. Furthermore, by specifying the objectives , and (and weighting them), users can control for model overfitting. Nevertheless, we are aware that if users do not specify , they may produce overfitted models. However, this may be the case for command-line classifier construction as well if addition of regularizers or cross-validation approaces are not used.

Scalability: The current UI design is based on datasets of modest size. Thus, there are aspects of the UI that would become less usable if datasets grew larger. For example, while the table view supports sorting and filtering the fundamental design of showing items in a table may make finding individual relevant data items challenging. However, for larger

datsets we can augment QUESTO with visualisations that can aggregate data and show example data items on demand (e.g., grouped heatmap view, etc.).

5.7 Summary

This chapter explained how users can communicate their objectives, constraints, and domain knowledge through the interactive construction of an objective function. Further, it showed the deployment of the technique in QUESTO, a VA tool which formalizes the mathematical embodiment of user preferences in the form of an objective function (a weighted linear combination of sub-objectives). Future iterations of this research might find solutions using a more complex representation of an objective function, i.e., facilitating a non-linear or quadratic objective function to evaluate model performance. Extending this interactive objective function technique further, the next chapter describes my proposed research work.

CHAPTER VI

CONFLICT DETECTION, RESOLUTION AND TRADEOFF ANALYSIS IN OBJECTIVE SPECIFICATIONS

Q4: *How can interactive visual interfaces help users to detect and resolve conflicts in objective functions?*

In multi-objective optimization, conflicting objectives and constraints is a major area of concern. In such problems, several competing objectives are seen for which no single optimal solution is found that satisfies all desired objectives simultaneously [103]. To that end, pareto optimal solutions are searched for, where each solution is better than the others in at least one objective [91]. Furthermore, visualization of these solutions may reveal not only the decision space but also may enlighten these conflicting objectives that obstructs selecting correct solution(s) for the desired task or goal. In order to make an informed decision, reconciling multiple conflicting objectives is one of the main challenges for these systems. For example, a user may emphasize to predict a set of relevant/critical data instances correctly, while mistakenly expressing that a subset of these data instances are outliers or noise in the data. To combat that, users often run multiple model simulations to explore the space of decision choices they have [25]. In our prior work QUESTO, the system supported showing multiple model options as potential solutions to users. However, QUESTO did not explicitly show model tradeoffs, conflicting objectives and failed to support users resolve these conflicts through it's visual interface.

Motivated to investigate tradeoffs in model selection and conflict resolution in interactive objective functions, I investigated the types of conflicts in objective functions. In addition, I looked at approaches to help users interactively resolve conflicts to create many variants of objective functions to understand tradeoffs in model performance (with respect to objectives). We prototyped a tool CACTUS, that simplifies user experience to specify meaningful objectives in interactive objective function based VA systems. This chapter explains: (1) CACTUS, a VA system facilitating exploration of model tradeoffs and helping users detect and resolve conflicting objectives, and (2) Evaluation of the proposed system through a quantitative and qualitative user study. These two goals seek to address the core challenges in objective functions: **R1** and **R4** as discussed in Chapter 5.

6.1 Problem Statement

Interactive specification of objectives and constraints may result in objective functions with conflicting objectives [109, 210]. Conflicting objectives or constraints may cause construction of inefficient objective functions that may confuse the underlying algorithm due to unclear user goals. For example, in a regression task, a user may specify to use a L2 regularizer to penalize attributes with large coefficients, but that may result into incorrectly predicting many relevant data instances, though improving the generalizability of the model. Here the objective to train a model with high accuracy on a set of important data items may conflict with the validation accuracy (or variance) of the model. Similarly, in a classification task, a user may expect to see similar data items in the same class labels, at the same time expecting that the global accuracy of the model is high for every class. The

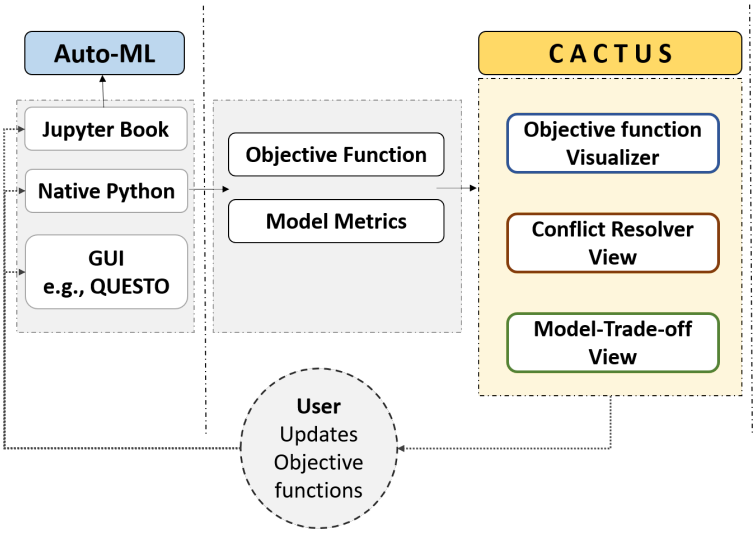


Figure 48: Workflow adopted in the system CACTUS.

model being trained to support the users request to place similar data items in the same label category, may not perform equally well for all class labels, thus dropping the global accuracy of the model. In the past conflicts in objective specification in multi-objective objective functions was addressed using tradeoff analysis [21, 109, 187, 253]. While useful, to our knowledge there is no other work in VA that have looked at resolving conflicts in user-specified objectives to build classification models using an interactive visual interface.

In this work, we extend research on interactive objective functions by helping users detect conflicts in objective functions, and further interactively resolving these conflicts to specify a more meaningful objective function to a model solver. To further understand what these conflicts are, and how adversely they may affect objective specifications, we conducted an extensive literature review. Grounded on the literature search, we enumerated a list of potential conflicts between various objectives in interactive objective functions explained later in the chapter. Furthermore, we present a visual analytic tool that facilitates: (1) Visualization of a multi-objective objective function that is defined using a python code (e.g, defined in a Jupyter notebook), or a visual interface such as QUESTO, (2) Highlighting conflicts between interactively specified objectives, (3) Helping users resolve these conflicts to adjust and improve their objective functions, and (4) Train multiple classifiers over time to perform tradeoff analysis of objectives in the objective function. As objective functions drive all ML algorithms, we consider visualizing objective functions may help users (e.g., novice ML users, ML experts who may use GUI to debug models, etc.) to understand their specifications to the underlying models, and further empower them to explicitly adjust the function terms to explore and test various hypotheses that they may have. More importantly, we consider supporting users in defining meaningful and correct objective functions is very important to ensure models that are sampled perform as per user expectations. With this approach, we seek to extend the current processes of interactive model tuning and model selection in which mathematical objective functions can be visualized and inspected. For example, in a Jupyter notebook one can define an objective function and then use our technique to visualize, test, and interactively adjust objectives to explore various model alternatives.

We prototyped CACTUS, a conflict resolution and tradeoff analysis system for user specification of objectives. CACTUS ingests an objective function (defined in Python or

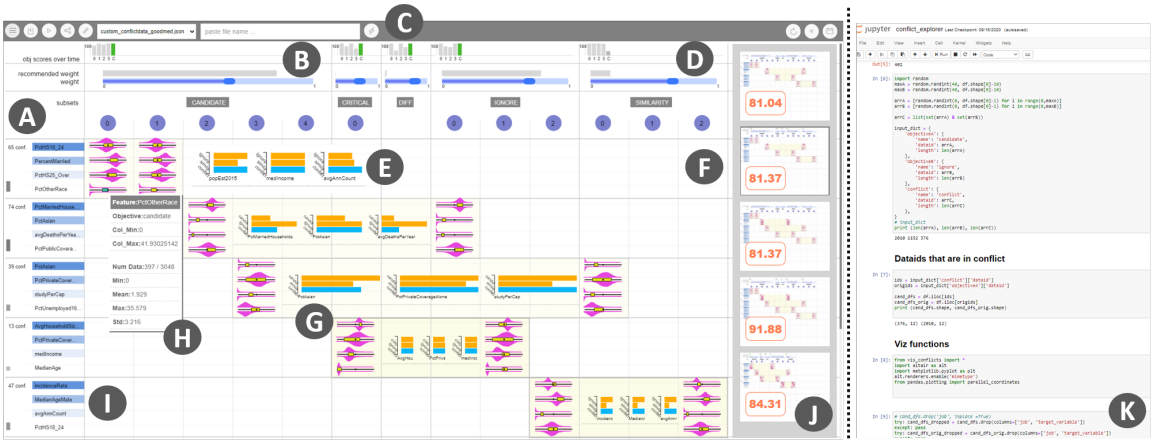


Figure 49: The CACTUS system - A. Data subsets per objective as seen in the header of the Conflict View. B. Model spark bars showing model performance per objective per iteration. C. Control bar to select objective functions and export objective functions. D. Gray bars show recommended objective weights. Blue sliders allow users to control weights per objective. E. Variance bars of top 3 highly variant attributes. F. Conflict View. G. Conflict box containing the violin plots with whisker boxes. H. Tooltip from a whisker box showing distribution of data on an objective. I. Top 4 highly variant attributes shown for each conflicts per row. J. Objective function gallery showing models' validation accuracy. K. Jupyer notebook, where objective functions can be defined.

Jupyter notebook) and then visualises it to show its objectives and respective weights. It visually explains the objectives that are closely satisfied by the selected classification model and the ones that failed to be satisfied (see Figure 48). Furthermore, it visualises conflicts between objectives, allowing users to incrementally improve the function by making adjustments interactively. When objective functions are adjusted interactively, users can re-train models, see a change in the models' performance and continue exploring the space of multiple variants of objective functions. Furthermore, visualising objective functions and the trained models performance metric (per objective) can empower users to explain/interpret and probe [84, 192] complex models in relation to how well they satisfy their goals. For example, users can probe if similar data items are predicted in the same class, or strong representative (candidate) data subsets are predicted with higher probabilities by resolving conflicts and creating multiple variants of objective functions.

In addition, we present the findings from a within-subjects user study. In this study we quantitatively evaluated CACTUS to test if it helped users to find and resolve conflicts, and then if it supported incremental training of models facilitating comparison of a varied set of objective functions. We also collected qualitative feedback to record user preferences in relation to easy of use, intuitiveness, and other relevant usability aspects of the system. Our study showed that: (1) Participants found CACTUS intuitive and expressive in visualising complex mathematical terms in a specified objective function for a classification task and also to find conflicts between objectives. (2) CACTUS helped participants ideate on multiple versions of objective functions and in the process resolve conflicts in objective specifications. We also present qualitative feedback from the participants that enlightens the strengths and weaknesses in the current UI design of the system, potential usability issues, and limitations that needs further research in the future. Our contributions are:

- An enumeration of potential conflicts in objective specification in multi-objective objective functions to construct classifiers.

- A prototype VA system CACTUS that visualises conflicts in objective functions and supports interactive resolution of these conflicts.
- A within-subject quantitative and qualitative user study validating that our technique helps users construct meaningful objective functions by resolving conflicts between objectives.

6.2 Types of conflicts in objective functions

In this section we describe types of conflicts in interactive objective functions. In order to understand what these conflicts are, we adopt the same objective categories as defined in QUESTO, e.g, *Instance-based*, (2) *Feature-based*, (3) *Train-objectives*, and (4) *Test-objectives*. Knowingly or unknowingly users may elicit various types of conflicts while specifying any of these objectives. We categorized these conflicts in the following groups:

1. **Conflicts based on choices:** Conflicts can be categorized based on users' subjective choices, presuming these choices follow best ML practices (e.g., guarding against overfitting, constructing classifiers that represent every class labels precisely etc.).

   Logic based conflicts: These are conflicts that are logically incorrect but does not violate best ML practices, which we term as *logic-based-conflicts*. For example, a user may specify a set of data items to be part of the objective *Similarity* (where these data items are expected to be in the same class label), while specifying subset of these data items as *Candidate* an objective representing a different class label (than the class label as annotated in the data). These conflicts are logically incorrect but do not violate best ML practices. Refer Figure 50 to see every conflict combinations.

   ML practices-based conflicts There can be conflicts which are not logically incorrect but may be violating best ML practices. For example, building a classifier using only *Train-objectives*, and not *Test-objectives* as a constraint, may produce overfitted models that are not generalizable to unseen data. Similarly for an imbalanced data if only an *Accuracy* objective is specified as opposed to *F1-Score* or *Precision*. This may trigger the model solver to select classifiers that performs poorly on minority class labels. In this work we are not addressing conflicts that may occur due to violating best ML practices.

2. **Conflicts based on time of occurrence:** *Logic-based-conflicts* can be further categorized based on when they occur in the modeling pipeline.

   Before model conflicts: Some conflicts can be computed before even training a model, while a few other conflicts can be only ascertained after a model is constructed. The first kind, which we term as *before-model-conflicts*, can be automatically computed before a ML model is constructed based on the objectives in the interactive objective function. For example, a user has specified a set of data items I that should be placed in the same class label, say Dog, while also specified a subset of I, say J data items as *Candidate* objective for the class label Cat. Here *Candidate* objective means data items that are strong representative of a specified class label. So the conflict is that the same data items are specified as examples of two different label categories Dog and Cat. Similarly another example of a *Logic-based-conflict* is between the constraints *Critical* and *Ignore*. While *Critical* represents the data items that are very important for the user, and thus the user expects the model to predict

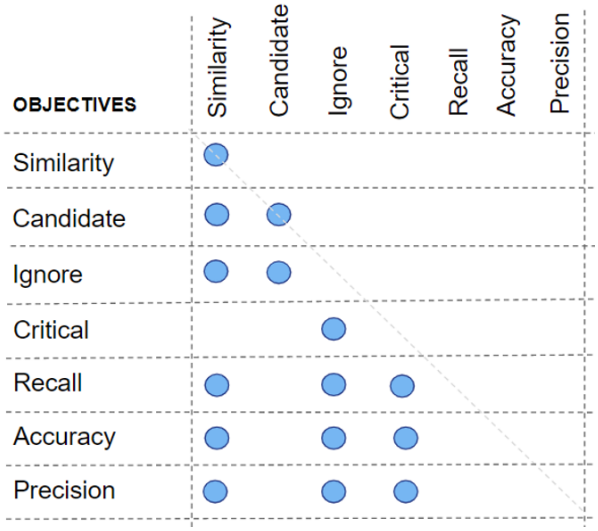


Figure 50: Conflict matrix, showing conflict possibilities between the objectives.

them correctly, while *Ignore* represents data items that are unimportant, or noise or garbage in the training set. A user may specify a set of data items as *Critical*, while in a future iteration of the model construction may specify a subset of *Critical* data items as *Ignore*. Such kinds of conflicts can be computed before a model is constructed.

After model conflicts: In addition to the above, there may be other types of conflicts that are only noticed when a model is constructed or many iterations of model construction have occurred which we term as *after-model-conflicts*. From the literature we know that in a multi-objective objective function, all objectives cannot be attained [109]. In such scenarios, a set of pareto-optimal solutions are presented to the user in which only a subset of objectives are attained in each of the pareto-optimal solutions [97]. Analysing model log data over multiple modeling iterations, the system can infer which objectives are repeatedly unattained, or which set of objectives cannot be solved together (at the same time). In such cases, these objectives are in conflict with one another, meaning that the specification of one, blocks attainment of the other or vice versa in the objective function [253]. For example, the system may infer that a highly weighted *Train-Accuracy* objective is prohibiting the model solver to find a model that also attains the *Similarity* constraint successfully. These conflicts can only be inferred when the model is constructed or when multiple iterations of model construction has occurred.

In this work, we scope our conflict detection and resolution method to handle conflicts that are logically incorrect and of the type *before-model-conflict*, but do not violate best ML practices.

6.3 Design Guidelines and Tasks

We have formulated the following design guidelines for CACTUS:

DG1: Visualize the objectives and the objective function. We seek to build a system that is able to visualize an objective function, showing its component objectives and assigned weights. Users should be able to visually perceive the function and understand what it means to the underlying model solver where it is ingested to sample classifiers.

DG2: Visualize model performance on each objective. Through our system we seek to show performance of the selected model on each of the specified objectives for every iteration of model construction. Users should be able to visually perceive how the model satisfied the specified objectives.
DG3: Show conflicts in various objectives. The system should visually show conflicts between objectives. Users should be able to visually understand the conflicts and seek details on each conflict (e.g., distribution of the conflicted data, see variance of attributes etc.).
DG4: Resolve conflicts: The system should help users interactively resolve conflicts. It should help them understand the conflicts in relation to the data items. Furthermore, the system should provide affordances to either perform conflict resolution through CACTUS or allow users to export conflicts to a Jupyter notebook (where users redefine objectives to improve the function).

Based on the above guidelines we set forth the following tasks that CACTUS should support:
T1: Import and visualize objective functions from a Jupyter notebook (NB). In addition, allow users to switch back and forth between multiple objective functions that they import through out their analysis process.
T2: Inspect conflicts between objectives to address potential problems with the objective function. Be able to compare severity of conflicts both between any objectives in a function, and between multiple objective functions.
T3: Resolve conflicts by re-assigning conflicted data items to a chosen objective. They should also be able to control implication of any conflict by adjusting weights assigned to these objectives. Users should be able to export conflicted data items' ID or the entire objective function to NB to further redefine the objective.
T4: Incrementally construct many ML models and inspect model tradeoffs between multiple model alternatives, through the interactive creation of objective functions. These functions can be variants of the function that they import created in the process of conflict resolution, re-assigning weights or re-writing the function in NB and importing again.

6.4 CACTUS: System Design

Here we describe the user interface and interactions supported by CACTUS. The main views of the system are: (1) Conflict view, (2) Model spark bars, (3) Venn diagram view, (4) Feature plots, and (4) Objective function gallery.

6.4.1 User Interface

Conflict view: This view shows conflicts using a matrix representation, where every column shows an objective from the loaded objective function (see Figure 49-F, **DG1**). The second row (with the numbered circles) represent subset data instances that were specified as examples as part of the respective objective. Next, every row in the table encodes a conflict (**DG3**) between a pair of objectives (e.g, *Similarity*, and *Ignore*). Conflict pairs are emphasized by a highlighted rectangular box (also called *Conflict box*, see Figure 51-H), where 4 most highly variant attributes are vertically ordered. Aligned with each of these attributes, a violin plot, with a whisker box plot is rendered (Figure 49-G and Figure 53). Here the violin plots show the distribution of the data in relation to that attribute, and the whisker plot shows the shape of the data instances that are part of the objective.

Further, users can hover their mouse on the whisker box to see detailed information about the shape of the data (Figure 49-H and 52-H). Thus, using this visual technique, users can compare the conflicted data instance's shape across two objectives that are in conflict with one another. The Conflict box, also shows a horizontal bar chart of *Variance bars*, where it shows the variance of the data in the two objectives (shown in orange) and the variance of the conflicted data items (shown in blue) in relation to top 3 highly variant attributes (see Figure 51-G). Using this view, users can understand how similar or different are the conflicted data in comparison to the data items that are part of the two objectives.

Model spark bars: On top of this view, CACTUS also renders horizontal sliders that allow users to interactively specify weights to the objectives (**DG2**). In addition, users can refer to system recommended weights, if they are unsure about the weights for each objective (Figure 49-D). CACTUS allows users to incrementally train models while they adjust the objective function (e.g., by resolving conflicts or updating objective weights). Per iteration when a new model is trained, its validation accuracy is plotted as a series of vertical bars, called *Model spark bars*, scaled between $0-100$ as seen in Figure 49-B). These accuracies are only on the objective of the column they are rendered in, thus allowing users to compare performance of the model with respect to specific objectives.

Figure 51: A. Objective function gallery shows previews of objective function per iteration with model validation accuracy. B. Model spark bars, green bars reflect current iterations performance improved over previous iteration. C. Gray bars show recommended weights. Blue bars are sliders to allow users to specify weights to objectives. D. Variance bars to compare conflicted data items' distribution with other objectives. E. Feature plots, blue dots are full training set, red dots are objective or conflicted data items. F. Context menu to resolve conflicts. G. Venn diagram view showing conflicted data overlap between objectives. H. Conflict box opens the bottom drawer to show venn diagrams and feature plots.

Venn diagram view: As users are exploring the conflicts, they can click on a *Conflict box* to trigger the system to open the bottom tray. It reveals a venn diagram showing the overlap between the pair of objectives. The size of the circles reflect how many examples comprise each objective, while the overlap of the circle encodes the conflicted data items

(see Figure 51-G). Furthermore, users can hover over the intersected region to see the distribution of the data on the *Feature plots* (Figure 51-E) and also on the *Variance bars* (Figure 53). Based on their exploration, they may decide to resolve the conflicts (**DG4**) by either: (1) Moving the points to the left objective or to the right objective, (2) Exporting the conflicted data to the Jupyter notebook, and (3) Completely removing the conflicts from the objective function using the context menu seen in Figure 51-F. Resolving any conflict, removes the visual representation of it, i.e., a single row from the *Conflict view*.

Feature plots: This view plots a set of k attributes with highest variance as small multiples of scatterplots (Figure 51-E). Each of these charts are plotted with the target variable (or dependent variable). The design of this view is intended to show users the shape of the data in an objective or in a conflict in relation to the input data (**DG4**). This view is linked with the *Venn diagram view*.

Objective function gallery: As users resolve conflicts or change weights of objectives, a new version of the objective function is stored in the memory. Users can access the history of objective function creation through this view, where each state of the objective function is shown using a thumbnail preview (see Figure 49-J). The preview also shows the trained models' validation accuracy score (between $0 - 100$) in this view (**DG2**). This view also supports users to go back to a previous state of the function, if the model performance drops.

6.4.2 Conflict Detection and Resolution Method

Next we explain how CACTUS detects conflicts and visualizes objective functions in relation to the found conflicts.

Conflict Parser: A user may write an objective function, O in Python/Jupyter notebook comprising of a set of k objectives $\omega_1, \omega_2, \omega_3, ...\omega_k$,. Furthermore, each of them can be specified with a set of k scores $(_1, _2, _3, ...k)$. Thus, O can be represented as a weighted linear combination of these objectives as seen here, $O = s_1 * \omega_1 + s_2 * \omega_2 + s_3 * \omega_3 + s_4 * \omega_4 + s_5 * \omega_5$. A objective ω_i is represented as a set of training data instance T ID's as $t_1, t_2, t_3....t_l$. However, in the case of *Candidate* objective, they may also be stored as ID's for a specific class label $L_{1...b}$ (b class labels). The conflict parser module of CACTUS checks for any overlap between all the paired combination of objectives from O.For example, it utilises T_i and T_j between the objectives ω_i, ω_j, using the function $FN(\omega_i, \omega_j)$ to find conflicted data ID's T_c. Finally, this module generates a hashmap object F, where keys are hashed to represent the objective pairs $(\omega_i - to - \omega_j)$, and the values are conflicted data ID's T_c.

Data Distribution and Variance: Conflicts are visualized using the F object (generated by the *conflict parser* module). Sequentially, the system tracks the pair of objectives P_i that are in conflict using the hash-keys (f_k) of the object F. Next it retrieves the set of data ID's T_i that are specified as examples as part of the objective pairs P_i. It also recovers T_j from F that represents the conflicted data items. Using T_i, the system first retrieves the top 3 (can be changed) attributes with highest variance in this set. Next it draws the violin V and the whisker W plots. While V shows the distribution of the full training set, the whisker plot W, shows the distribution of the examples that are part of the respective objectives in P_i. Similarly, the *variance bars*, are rendered to visualise the variance for the data ID's in P_i, and the conflicted data items T_j. The venn diagrams are also drawn using the object F.

Model Solver: Users can also utilise any Auto-ML model solver M to sample n classifiers C (e.g., can be adjusted $n = 200$) in Python or in a Jupyter notebook. For the current prototype we tested with Hyperopt [127], but can be replaced by Auto-SKLearn [77], or Auto-Pytorch [], if required. In this pipeline, M expects an objective function O, to score each of the classifiers c_i in C. The highest scoring (H) classifier c_k is selected and the performance (accuracy score) overall, and per objective is visualized in CACTUS. More importantly, model construction is handled by the Python code or the Jupyter notebook, CACTUS only ingests the objective function O, and visualizes conflicts.

Weight Recommendation: The set of weights $S = s_1, s_2, s_1....s_k$ in O can be specified from the Jupyter notebook. These can also be interactively adjusted from the interface of CACTUS using the sliders (Figure 51-C). To further guide users, our approach also recommends weights S' (between $0 - 1$, for each objective ω_i). In the initial iterations, the recommendations are randomly initialized, however, as users incrementally construct multiple versions of objective functions ($O = O_1, O_2, O_3, ...O_f$) and models ($M = M_1, M_2, M_3, ...M_f$), these weights are recommended based on the probabilistic likelihood of the weight settings that found success in: (1) Maximising the overall model accuracy, and (2) Maximising the objective score for which the weight is recommended. We followed the bayesian approach [77] in modeling the probabilistic likelihood of the weight settings.

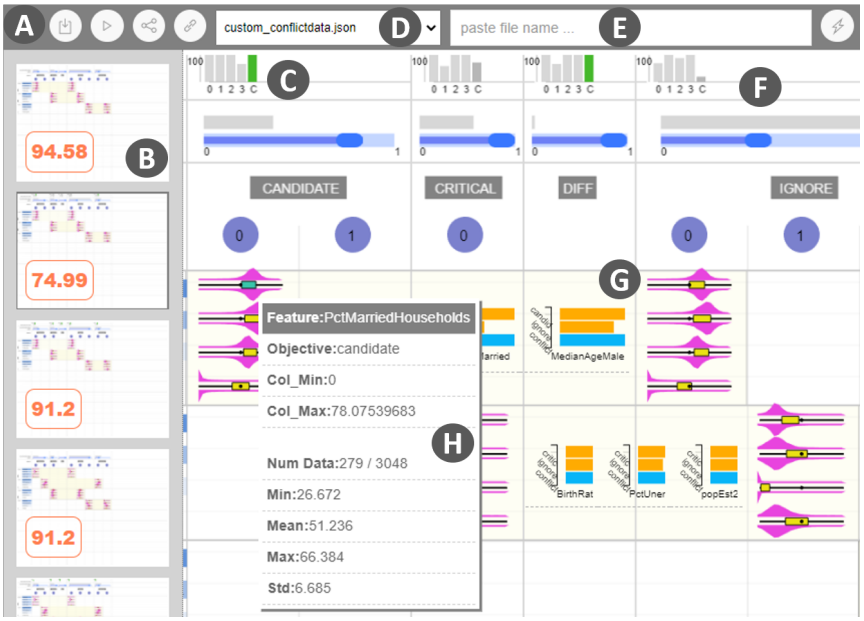


Figure 52: A. Buttons to train models, load and generate objective functions. B. Objective function gallery. C. Model spark bars encoding validation set accuracy. D. Drop down objective function selector. E. Specify path to an objective function to load in CACTUS. F. Objective weights. G. Conflict box and Variance bars. H. Tooltip seen on mouse hover over whisker boxes.

6.5 Case Study

Here we describe a case study, where a user works in Jupyter notebook (NB) along with CACTUS to construct a classifier. In this process, they design multiple variants of an objective function and then visually inspect their effectiveness. Using CACTUS's conflict resolution technique, they discover and resolve conflicts in the specified objectives in the

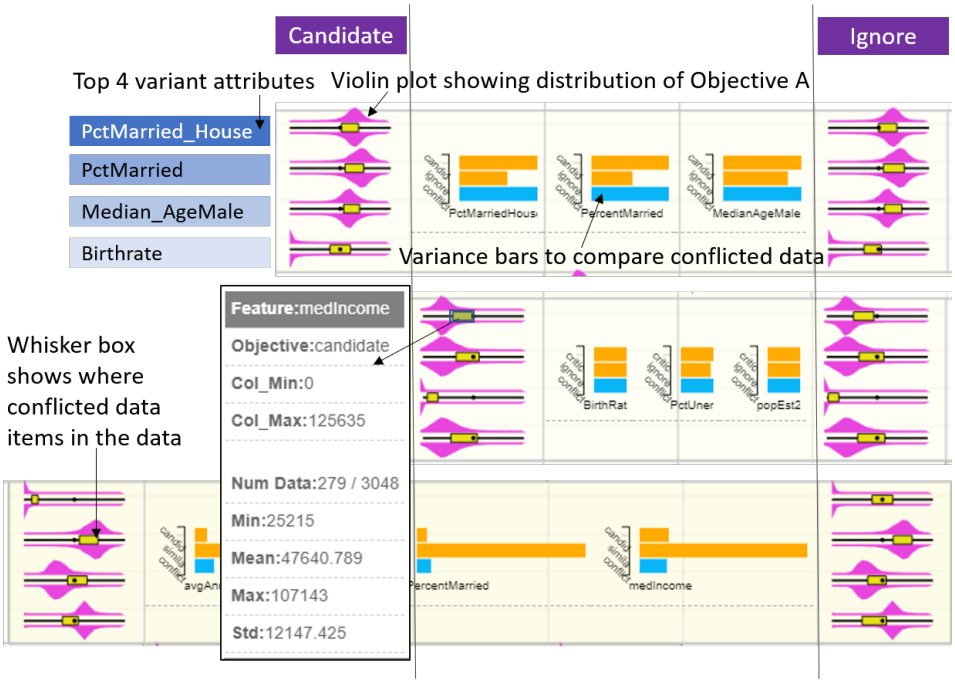


Figure 53: Violin plots and whisker boxes show how conflicted data items are similar or different to the objectives.

objective function. Consider James is a data analyst in the public policy department of USA and seeks to construct a classifier to predict *Cancer mortality rate*. They have access to a *Cancer mortality dataset (per US County)* [**?**] containing 3048 records (rows in the table); each row represents a US county. It contains 34 independent variables such as *incident-rate*, *median-age*, *avg-household-size*, *birth-rate*, *perc-resid-health-coverage*, and others. James wants to predict the *Death-Rate-Per-County*, containing labels: *negligible*, *low*, *medium*, *high*, and *very-high*.

Define and visualize objective functions: James uses NB to explore and analyse the data (Figure 49-K). To construct a classifier they first partition the input data U into three sets: (1) Training set R (2300 samples), (2) Validation set S (600 samples), and (3) Test set T (100 samples). They further partition the validation set into multiple subsets ($S = S_1, S_2, S_3...S_i$). Next, James writes a Python class object to construct a gradient boosted model to classify the data. When trained using this model, James observes a relatively poor accuracy of only 72%, and 65% on the training and validation set respectively. Motivated to select a preferred optimal model for this problem, James decides to use Optuna - a hyperparameter tuning/Auto-ML solver, and writes a custom objective function for this package to solve for [9]. James defines an objective function (also called loss function) in NB containing a set of objectives such as *Candidate*, *Ignore*, and *Critical*. At this point James re-trains a new classifier by feeding this objective function in Optuna, and notices that the training accuracy improved to 86%, while the validation set improved only marginally (73% accuracy). In their objective function specification James filters the data with less than value 50000 of the attribute *med-income* and higher than 18% of the attribute *poverty-percent* to be classified in the same class label of *medium*, as they are similar to one another. Furthermore, they select a set of data samples F whose $birth - rate$ and $median - age$ attribute values range between $6 - 15\%$ and $35 - 50$ respectively, as critical counties that

should be correctly classified. Thus in the objective function, James specifies a condition to penalize any sampled model if it makes a mistake in predicting any of the data items in F.

Discover and remove conflicts in objectives: Motivated to explore different variations of this objective function and to validate any conflicts between the specified objectives, James imports it in CACTUS (on a separate tab in the browser, Figure 49). James sees the conflicts in the *Conflict view*, and looks at the left end of the row to find the most severe conflict (Figure49-A). They see that there are 39 data samples that are in conflict between *Similarity* and *Candidate*. They also inspect the conflicted data items' position with respect to the attribute values shown through the violin plots and whisker boxes as seen in Figure 53. James notices that the examples shown as part of the objectives have similar shape and thus likely to be confusing the model solver. They click on its *Conflict box* to reveal the *Venn diagram view* (see Figure 51-G) showing the conflicted data samples. Next to this view, they explore the *Feature plots* to see the data points in each of these objectives, along with the ones that are conflicted (Figure51-E). Hovering over each of the venn diagram's arc sectors, James inspects the data samples on the *Feature plots* to answer if these samples are similar or different from the two objectives (as seen in the red dots in Figure 51-E). James realizes that a condition to specify a set of data samples with attribute values of *med-income* lesser than 50000, might have caused this conflict. They export this conflict to Jupyter notebook (NB) to further re-define the objectives *Similarity*, and *Candidate*. In NB James adds two more objective subset examples for: (1) *Similarity*, and (2) *Critical*. Using the *Candidate* objective, James specifies a set of exemplary counties that are good examples of the class label *high* for death-rate prediction. James trains a new model and exports the new objective function to CACTUS for further analysis.

Iterative modeling by conflict resolution: CACTUS updates the *Conflict view* with the new objective function. James looks at another conflict between *Candidate* and *Ignore* (see Figure 49). They click on its *Conflict box* to inspect the venn diagram. They understand that 74 examples specified in the class *medium* for *Candidate* are actually from different class labels. In addition, they see a subset of the *Candidate* examples are also specified as part of the *Ignore* objective. To test the models' performance without a conflict between *Candidate* and *Ignore*, they reduce the weight on the objective *Candidate* to 0. James trains a new model using this weight setting (Figure 51-C). CACTUS visualizes the newly trained classifiers' metrics on the *Model spark bars* on top (Figure 51-B). and overall models' validation set accuracy in the *objective function gallery* to show James notices the model has a *Validation-accuracy* score of 80.1% (Figure 49-J). and further understands removing the objective (by setting its weight to 0) marginally helped the models' performance.

Next, James expects to improve the classifiers performance further. They first export the objective function and the conflicts to NB and then inspect the model there in relation to a set of counties from the training set (using the buttons seen in Figure 52-A). They find many important counties that are incorrectly classified by this model. James re-defines the *Similarity* and the *Candidate* objective to take note of it. From CACTUS James observes that one of the most severe conflict was between *Critical* and *Ignore*. To remove the conflict between them James filters the conflicted data samples using the data ID's retrieved from CACTUS. They find similar data samples (using cosine distance) to the conflicted data samples from the training set for the *Critical* objective. Next, James trains a new classifier, loads it in CACTUS to see that the *Validation-set* accuracy improved to 94.58% (Figure 52-B). Happy with the analysis results and the models' trained James

exports the model and the final objective function to share with their collaborators. In this usage scenario, James uses Jupyter notebook and CACTUS hand-in-hand, to design objectives and remove conflicts respectively. The workflow helped them select a classifier that optimally performs on specified objectives.

6.6 Evaluation

We conducted a qualitative and quantitative user study of CACTUS to validate the effectiveness of our conflict resolution technique. As we did not find any other visual analytic system system that helps users to find and resolve conflicts in interactive objective functions, we could not design our study to compare results and prove statistical significance of any measure. Thus, given the constraints, we designed our study to address the following research questions:

RQ1 Does CACTUS make it easy to precisely find conflicts between objectives in objective functions for classifiers?

RQ2 Does CACTUS support users in correctly resolving conflicts between objectives?

RQ3 Does CACTUS help users to compare objective functions and understand tradeoffs between them?

We recruited 14 participants (9 Male, 5 Female), aged between $22-36$ ($M = 26.06$ $[22.41, 29.71]$), by inviting participants through our university mailing lists. Our requirement was that they should know how to read/write basic python code, with elementary understanding of classifier construction and exploratory data analysis. Our participants were a mix from masters and PhD students from computer science, analytics, geography, and urban planning. They had basic familiarity with data analysis ($M = 5.26$ $[3.73, 6.79]$, on a Likert scale rating of $1-7$, higher is better), and basic ML expertise ($M = 4.85$ $[3.63, 6.07]$). The study was conducted completely remotely using Bluejeans [1]. It lasted 60-70 minutes and at the end of a successful session we compensated participants with a $10 Amazon gift card. The system was deployed on our computer, which we shared using a publicly accessible URL retrieved using NGROK [2], to conduct the study.

6.6.1 Study Design

We began the study by using a live demo, showing participants how CACTUS works and how its various visualizations can be interacted with We also demonstrated how the system integrated with a Jupyter notebook environment to seek objective functions (pre-defined by writing Python scripts), and data instances. During this session, we encouraged participants to ask as many questions they wanted to clarify any confusion with respect to the workflow or the system interface. Next, when we felt confident that participants were ready for the tasks, we prcoceeded to the experimental sessions. To answer the previously mentioned **RQ**'s we considered these dependent variables: (1) *Task completion times* to detect/find and resolve conflicts, (2) *Conflict resolution success rate*, i.e, the number of conflicts the participants correctly resolved out of the total conflicts for all the given objective functions (between $0-1$) etc., (3) *Model Accuracies*, accuracy score of models per iteration

[1] https://www.bluejeans.com/
[2] https://ngrok.com/

of objective function specification (between $0-1$), (4) Number of iterations as users incrementally created objective functions by resolving conflicts, and (4) *User preference ratings* that includes *Ease of use*, *Intuitiveness of the GUI*, *Steep learning curve*, and other relevant system interactions (all of the scores were normalized between $0-1$).

6.6.2 Datasets

For the practice session, we provided a dataset of 5000 IMDB movie records [3]. The data had attributes such as *gross-revenue*, *budget*, *cast-facebook-likes*, *number-user-votes*, etc. It was a ulti-class classification task to predict the rating of a movie between *low*, *moderate*, *high*, and *very-high*. For the first experimental session, we provided San Francisco city's employment dataset [165] containing 25000 records of job types for the quantitative evaluation. Each row in the data contains information about a job's remuneration information containing attributes such as *dental-benefits*, *annual-salary*, *health-benefits*, *retirement-compensation*, etc. The task was to predict the job's department which had 5 classes e.g., *Cultural/Recreation*, *Public Service*, *Healthcare*, *Administration*, and *Other*. For the next experimental session, we provided the *Cancer mortality dataset (per US County)* [**?**] to predict *Death-Rate-Per-County*. The class label were Very high, *High*, *Moderate*, *Low*, and *Negligible*. The dataset contains 3048 rows, each row representing the death rate of a US county and falls under one of the five categories of class labels. Furthermore, it has 34 attributes (1 categorical variable) including *incident-rate*, *median-age*, *avg-household-size*, *birth-rate*, *perc-resid-health-coverage*, and others.

6.6.3 Tasks and Procedure

In the practice session we provided users with a list of 3 pre-defined objective functions (written in a Jupyter notebook) on the IMDB movies dataset [3]. We asked them to load each of the objective function in CACTUS and visually explore the conflicts in various objectives. Next after 15 minutes of practice we asked them questions such as: (1) Which objective pair has the highest conflict? , (2) Name the top 2 highly variant attributes for the conflict between *Ignore* and *Similarity*, (3) Resolve conflicts between *Similarity* and *Candidate*. (4) Export conflicts between *Candidate* and *Ignore* to the Jupyter notebook, and (5) Adjust the weights of the objectives and train a new model on each of the given objective functions, and then export the best objective function based on model accuracy score. When we ensured they understood the concept and the interactions supported by the system, we moved on to the experimental sessions. For session *A* we asked participants to load three objective functions (pre-defined by us in a Jupyter notebook) on the San Francisco's salary dataset [165]. We asked them to perform the following tasks:

Task 1 Report number of data items that are in conflict between *Ignore* and *Candidate* from the second objective function.',

Task 2 Name the top 2 attributes with high variance between the objectives *Similarity* and *Candidate*.

Task 3 Resolve the first and the last conflict from the third objective function. Train a new model after you resolve each conflict and then compare the model performance. Export the objective function with better model accuracy.

Task 4 Out of the three objective functions, find the objective function that has the highest conflict between any of the objectives.

Task 5 Train three models by changing weights of any of the objectives for each of the three objective functions. Which objective function found the better performing model?

Task 6 Export any conflict from the the top 2 best performing objective function from your list of saved objective functions, to Jupyter notebook.

In the next session (B) we asked participants to freely use CACTUS using a given predefined objective function and a given baseline classifer on the Cancer mortality dataset [**?**]. There task was to:

Task 7 Incrementally improve the baseline models' accuracy using any of the interactions supported in CACTUS in 8 minutes.

In total participants performed 7 tasks using 2 datasets to build a set of classifiers. To remove any learning effect, we randomised the tasks across participants.

6.6.4 Data Collection

We captured video and audio of participants screen while they interacted with CACTUS. We saved log data which stores (per iteration) models' selected by users, their learning algorithms, and hyperparameters, predicted class labels, interacted objectives and conflicts, etc. When participants completed all the tasks for both sessions, we asked them to fill a NASA-TLX form [93], and a post-study questionnaire with a set of Likert scale questions (7 point scale). In the end we conducted a semi-structured interview asking open-ended questions about the workflow, system usability, and interaction design for each interface. For example we asked: (1) Explain your strategy to resolve conflicts? (2) Elaborate your thoughts on CACTUS's workflow, design and interactions., (3) How can we improve the current design of CACTUS? Through out the tasks we encouraged participants to think aloud while they interacted with CACTUS. Next we present results from the study both quantitatively and qualitatively. In any of the analysis presented below, we did not compare the results with any other system/tool, as to our knowledge, we don't know any visual analytic system that supports conflict detection and resolution in objective functions for any ML task.

6.6.5 Quantitative Analysis

We broadly wanted to measure if using CACTUS: (1) users can detect conflict easily and successfully, (2) users can resolve conflicts with precision, and (3) users can compare objective functions over time and learn tradeoffs between them. Thus, we measured CACTUS's success based on the following quantitative metrics:

Task completion times: We measured task completion time when users were asked to: (1) Report a conflict between a pair of objectives ($M = 2.43mins.$ $[1.41, 3.45]$), (2) Report highest conflict between the three objective functions ($M = 5.12mins.$ $[2.81, 7.43]$). Next, we measured task completion time when participants: (1) Resolved conflicts between a pair of given objectives in an objective function ($M = 3.03mins.$ $[2.59, 3.47]$), and (2) Resolved conflicts across the three objective functions ($M = 5.02mins.$ $[4.00, 6.04]$). The relatively

lower task completion time (in comparison to writing code to do the same task) answered **RQ1** that CACTUS makes it easy for users to succesfully find and resolve conflicts.

Correctness in conflict resolution: In addition, 13 out of 14 participants successfully found the right conflicts between objectives. Thus answering **RQ1** we observed 92.86% success rate among participants to find conflicts in an objective function. We also observed that every participant was able to successfully resolve conflicts (100% success rate). However, we found that in the case of two participants, while they succesfully performed the interactions to resolve conflicts, the system failed to correctly record the specified changes and thus failed to update the objective function. This can be attributed to either of these reasons: (1) a bug in the system, (2) a latency in the network, as the study was conducted completely online due to the on-going COVID pandemic, or (3) the interface did not visualize the correct objective function data after the conflict was resolved. In future, we will look in to this issue further to understand the most likely cause. However, given the high success rate of the task ($12/14 = 85.71\%$), and the quick task completion time relative to writing programs or codes in Python or R, we consider these measures answer confirms **RQ2**.

Incremental comparison of objective functions: To answer **RQ3**, we further measured log data to assess if participants were able compare objective functions and learn tradeoffs between objectives as they trained multiple classifiers. In doing so, we observed that on an average, participants iterated 10.23 times ($M = 10.23$ $[6.69, 13.77]$) to improve the given classifiers' baseline accuracy score of 78.24% on the Cancer mortality dataset [**?**]. 11 out of the 14 participants selected an objective function (to export, as their final selection) from a previous iteration in time. Furthermore, we observed two approaches to train new models: (1) Train a new classifier by changing weights only (2/14 participants), (2) Train a new classifier by resolving conflicts only (4/14 participants), and a hybrid approach of the two which was the most popular (8/14 participants). We also measured to find that 88.34% of the participants found success in improving the baseline classifiers' performance.

User preference ratings: We analysed Likert scale user preference ratings (on a scale of $1-7$, higher is better) provided by users after they interacted with CACTUS. Participants expressed that CACTUS was *Easy to use* ($M = 6.04$ $[5.85, 6.23]$), and it's interface was *Intuitive* ($M = 5.51$ $[5.05, 5.96]$). The majority also confirmed that the interface did not have a *steep learning curve* for new users ($M = 2.45$ $[1.25, 3.65]$), lower is better in this case). Most of the participants felt that they were successful in resolving conflicts ($M = 5.11$ $[4.95, 5.27]$), and they were able to incrementally improve models' accuracy score ($M = 5.23$ $[5.01, 5.45]$). Furthermore, participants confirmed that the *Conflict view* was expressive to not only help them find conflicts but also know which conflicts were more severe from the set ($M = 6.03$ $[5.54, 6.52]$). Likewise the *Venn diagram view* was found to be very useful to resolve conflicts ($M = 5.32$ $[5.11, 5.53]$). However, majority of the participants found the *Feature plots* were not as effective to resolve conflicts ($M = 3.11$ $[1.55, 4.67]$). Next, from the NASA-TLX survey, we observed that on average every participants' mental workload, and frustration towards the tasks were on the lower side ($M = 2.11$ $[1.01, 3.21]$ out of a 10 point scale; lower is better). Based on these findings, we are encouraged to continue research along the lines of interactive objective functions. However, the qualitative feedback enlightened other aspects of the system that may need further improvement or redesign in the future.

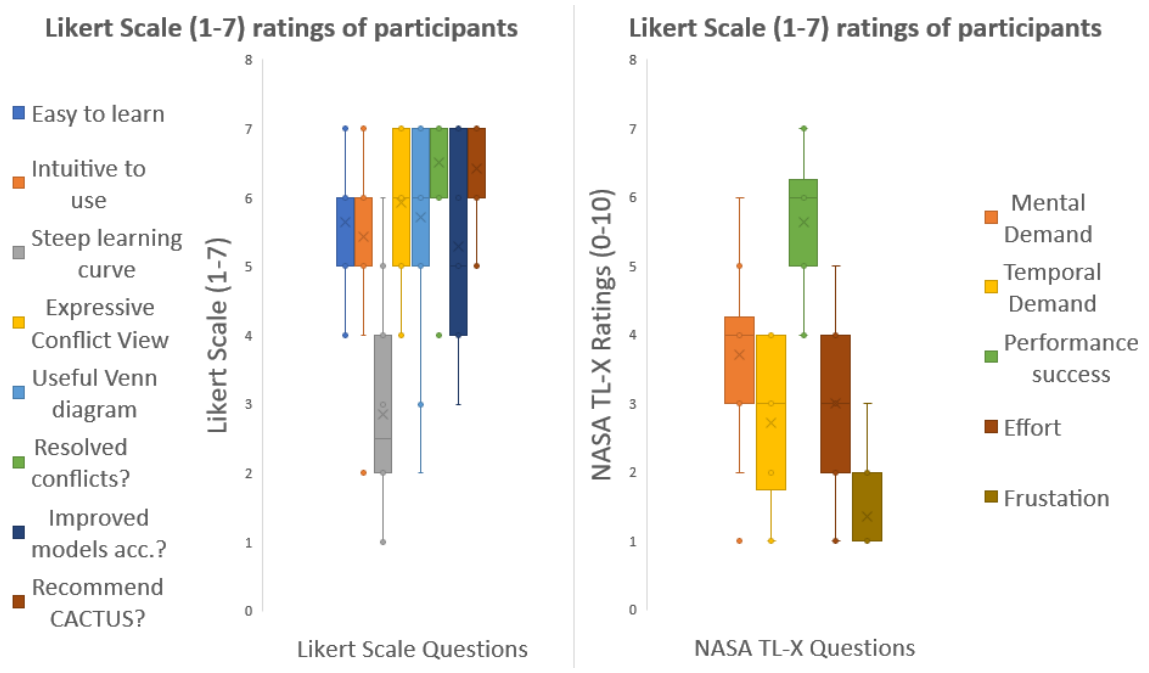


Figure 54: Study results of Likert scale ratings and NASA TL-X scores for CACTUS.

6.6.6 Qualitative Analysis

Expressive visualisations and intuitive interactions: Most of the participants liked the visual representations used in CACTUS to represent conflict. In addition, their feedback confirmed that they liked the current interaction affordances the tool suppports to help them interactively resolve conflicts. **P10** noted, *"Easy to use, ui was straight forward. Resolving conflicts was easy. I liked that I could do things without much interruption and whenever I wanted the most."*. However, **P07** pointed *"I could easily find most severe conflicts, which was also my approach to resolve conflicts as I trained model. But, the sorting of the conflicts could be based on some rationale (eg. number of conflicts, or other criterias, etc.) to prioritise urgent conflicts first."*. This feature is relevant and we plan to incorporate in the UI in future.

Incremental objective function exploration: We observed that participants enjoyed the workflow of incrementally changing objective functions to train better performing models. They either changed weights or they resolved most severe conflicts to create a variety of objective functions and trained models. **P04** expressed *"I liked the plot on the top that shows you the history of the model performance, it was very helpful for me to keep track of the model improvement."*. Few participants gamified the model training process, with the goal to train models that perform better than the models performed using weights recommended by the system. **P03** said *"I liked the highlighted bar chart view with the attributes with the highest variance. That helped me tweak my weights as well as resolve conflicts, and in doing so I was trying to beat the machine to find a better model"*.

Glean insights about the process: A few participants expressed the desire to learn a bit more about the process, in addition to be able to interactively train models. For example, **P02** shared *"At times it was hard to know how to change the weights and whether to move*

left or right when trying to resolve conflicts. I expect to see more visual cues on how to improve models' accuracy.". Similarly, **P01** noted *"Doing things were easy, but doing them well took more background knowledge than I had. It would help to know how the function was driving the modeling process"*. In this iteration, we developed CACTUS as a proof of concept prototype that confirmed that conflicts can be detected and resolved interactively, however, in future, we plan to making the incremental modeling process more transparent using GUI elements. In the current setting, users can learn more about the modeling process by exporting the objective function or the conflicts to the jupyter notebook environment.

Conflict resolution strategy: Broadly, we understood two main strategies participants used to resolve conflicts once they found which conflict to resolve. The first approach was to compare the shape of the data in conflict, to the data in the two objectives by observing the *Feature plots*. Based on the shape resemblance they moved the conflicted data instances to one of the objectives. The other approach was to look at the **Variance bar view** to find resemblance of the conflicted data to one of the objectives. When participants were not sure, how to resolve the conflict, they preferred to export to Jupyter notebook to provide better examples by writing Python code. **P08** said, *"My first step would be to find which variables would be useful to analyze in the conflict view. I would then, look at the bar charts to see if the conflicts were closer to one of the objectives. If not, I would look at the scatterplot and make judgements by approximating the average location of each of the three groups."*. In the future, we plan to provide more visualization supports to help users decide on how to resolve conflicts.

6.6.7 Study Limitations

The tasks are designed so that they can be completed within $60 - 70$ minutes. However, we understand that in real usage, these tasks can be more exploratory and may take longer times. To construct classifiers we specified a hand chosen list of hyperparameters to Hyperopt Auto-ML, with values that can be samples from a specified domain range. The study results may get affected if a different Auto-ML tool is selected or a different set of hyperparameters are tuned. The scale of the data that we used are medium-sized ranging in thousands (< 10000 samples). The results may vary if we use large datasets (e.g., $100k$ or more). Due to the COVID-19 pandemic, we had to conduct the study completely remote, which has its own set of limitations in comparison to in-person controlled lab study settings such as lack of direct observations, network latency issues, in-adequate feedback from think-aloud protocols etc.

6.7 Discussion and Limitations

Exploratory model space analysis: Enabling users to interactively construct multiple variants of objective functions empowers them to explore the model space (set of models defined by their unique hyperparameter settings) in an ad hoc iterative workflow. Users can adjust weights of objective, resolve conflicts, redefine objectives in Python and then import in CACTUS to visualise new objective functions. These functions trigger Auto-ML to search for models that are better aligned with their goals. The studt further confirms this hypothesis that CACTUS encourages exploratory model space analysis using objective functions as a means to communicate user preferences to find optimal solutions to their ML task. Furthermore, we observed that visualising conflicts and model performance scores with respect to specific objectives, helps users interpret implication of the selected model

on the expectation of the user. In some way, we can assert that objective functions support interpretation of models' performance, however, we understand more research needs to be done to confirm if interactive objective functions can further explain models.

Collaborative objective functions and conflicts: From our experience with objective functions, we realised that in real world ML teams, objective functions are often designed by multiple members of the team to solve the same problem. In situations like this CACTUS can be a tool that helps compare multiple versions of objective functions visually, from different team mates, and then explore the space of possible objective functions and conflicts that may arise from each. That may open a new research area as an extension to our current work, in which users may like to resolve conflicts by collaboration across different geo-locations, or from different devices (e.g, from browsers accessing CACTUS synchronously and resolving conflicts). We plan to look into this area of work in the near future.

Gamification of modeling using objective functions: In the study we observed few participants, amified the process of model construction, by testin various weight settings of objectives, resolving conflicts, or specifying new objectives. Their goal was to beat the previous models performance score and also to perform better than the model constructed using system recommended weights. Many participants elicited there is a lot of value in this, as not only it helps them ideate and explore faster, but also makes the process of finding a suitable model more fun. We realise incremental construction of objective functions, and being able to revert back to any time step in this process augments them to freely explore and experiment with their hypothesis. In future, we see lot of potential in supporting the interface with features that further encourage and guide users in this process of gamified model creation.

Current limitations in conflict resolution: Though CACTUS allows users to incrementally create and compare objective functions in the process of resolving conflicts, through the study we found many aspect of it needs further researcha and work. For example, we found the current view showing the time step view of objective functions (*Model spark bars* and *objective function gallery*) is at times difficult to track because it only supports sequential record. Every change in one parameter leads to a model retrain, and saves a new copy of the objective function. This may take some interaction time and may be difficult to track when users have iterated many times. We aspire to research further to find potentially better design choices interms of visual design and interaction to present time line representation of models and objective functions.

Another issue, we realise is that data exploration is critical to resolve conflicts and design good objective functins. A few participants in the study, who were less versed with the data, felt very difficult to know how to change the weights and whether to move left or right when trying to resolve conflicts. Sometimes this process felt like trial and error. In CACTUS we focussed on conflict resolution and objective function comparison, while separated the task of data exploration in Jupyter notebook. While that may work for advanced users with Python and Jupyter notebook experience, it seemed for intermediate users, an aspect of the interface should support data exploration in the future.

Scalable conflict detection: The current prototype's implementation is designed to be model agnostic, meaning it can work with classification, or regression models (supervised ML). It can also work with sequence data such as text, time-series and image data. Currently, the interactions are tested on medium sized data sets (i.e., with data samples in the

range of 100k). The design of the visualisation uses Phaser JS (a 2D game engine library for the web) with WebGL on canvas implementation. In the next iteration the conflicts can be ordered, grouped and can be extended to represent more than two objectives per row in the table. Though we tested with multiple conflicts per row, we discarded the approach, as it may lead to ineffective selection of conflict intersections in the venn diagram view. In future, we wish to research potential solution for multi-objective venn diagram based solutions to represent conflicts.

6.8 Summary

Through this work, we present our research on making interactive visualizations of objective functions in the process of constructing machine learning models. Furthermore, with our visual analytic system CACTUS, we demonstrated a novel technique that interactively resolves conflicts among objectives in a multi-objective objective function. In CACTUS, we also discussed an enumeration of various types of conflicts that may occur when users specify objective functions to ML model solvers to classify tabular data. With a quantitative and qualitative user study we show that our technique helps users to interactively visualize objective functions, resolve conflicts between objectives, and incrementally train ML classifiers in tandem with a Jupyter notebook environment. In future, we are motivated to continue our research on interactive objective functions to guide users explore and ideate multiple variants of these functions as they collaboratively solve analytical problems.

CHAPTER VII

DISCUSSION

Machine learning (ML) has changed various problem domains by offering insightful solutions such as biologists clustering genome samples, medical practitioners predicting the diagnosis for a new patient, ML practitioners tuning model hyperparameter settings etc. To support users to freely incorporate machine learning in their domain there has been a recent surge in making machine learning interactive through applications that support a variety of data analytic tasks (e.g., H2O, OrangeML, etc.). These tools and algorithms support interactive construction of models for people with various expertise in ML; from non-experts, to intermediates, to advanced users. However, designing these systems/algorithms includes addressing numerous challenges, such as diversity in user expertise, metric selection, user modeling to automatically infer preferences, measuring success, etc. Through this research, I designed visual analytics systems to incorporate user interaction data in machine learning modeling pipelines. Specifically, I analyzed interaction techniques and ML algorithms by measuring user satisfaction ratings, success rates in finding user-preferred models, model accuracies, and task completion times. I found that these approaches empowered people to communicate their preferences to interactively construct machine learning models to support numerous data analytic goals (compared to Auto-ML tools). In this thesis I have demonstrated two main techniques in interactive modeling: (1) multi-model steering, and (2) interactive objective functions. This work facilitates specification of user goals and objectives to underlying ML model(s) using interactive visual interfaces on the web. Broadly, these techniques support the following aspects of this research:

1. **Novel interaction methods with multiple models:** Incremental construction of multiple ML models, and selecting a set of preferred model(s) is a very complex task that requires technical expertise in both software skills and elementary data science. For instance, current multi-model systems [37, 179] require users to be able to adapt to the steep learning curve of interacting with a complex user interface with a complex workflow in addition to be able to comprehend model metrics, hyperparameters etc. While a part of this research have investigated novel approaches of user interaction with multiple ML models [50, 225] to help users learn about them and the underlying data (in the process of interactive model selection), another part have addressed simplifying the complex user experience of interacting with multiple models prevalent in current systems.

2. **Interactive navigation of model space:** To simplify understanding the implication of a model on the input data, this research facilitated exploration and interactive navigation of the model space [225] in relation to prediction on relevant data instances [50, 54] or explaining a models' reasoning process through the visualization of top weighted features (by the model) [51] and their correlation with the target label in the data [52]. For example, if a user notices that a relatively average performing model weights features that are more relevant to their domain, they may feel more confident about it than a similar highly performance black-boxed model. This may motivate them to further computationally search for similar models from the model space.

3. **An innovative technique to interactively construct objective functions:** This research presents ideation, prototyping, and invention of a novel visual technique to interactively construct objective functions supporting classification tasks in ML. Furthermore, this thesis also explored explicitly showing trade-offs and conflicts in objective functions to users to support them design more effective and correct objective functions. This technique interactively allows creation of a diverse set of objective functions, to sample models that addresses a diverse set of user goals, through the lens of which users can learn trade-offs in objective specification as they decide on which model to choose.

7.1 Challenges in user preference specifications in interactive machine learning

Below I describe challenges that I faced throughout the research, some of which are still an open area for further investigation.

- **Computational scalability:** Multi-model based VA systems tackle computationally expensive operations; for example, users interactively train multiple ML models. When they are trained, the system presents the results (e.g., prediction per data item) to users. However, as models' complexity grows, so does the training time, which renders interactive real time feedback to the system a big challenge. Furthermore, in my research, this posed to be a bigger problem as the datasets size grew (from thousands of rows to few hundred-thousand rows). For instance, if conflicting objectives are to be identified before even starting the model training process, then the system must look into every data sample and their data values across all the dimensions to find objectives that may conflict with one another. For larger datasets with high dimensionality, this tended to be computationally expensive, and obstructed real-time interactions of the VA system. I scoped my research to address ML problems supporting medium sized datasets (i.e, 80 to 100 thousands sample dataset or less). In addition, in some of the presented systems, I followed a progressive visual analytics style approach [222] (showing intermediate results, while training models) with interactive web caching [40]. This helped to provide a real-time interaction experience to the user, with intermediate results as the computational task (model training or conflict resolution) continued in the background. However, in the future, these scalability related issues need further research to discover new approaches to ensure interactive modeling can further scale to larger sized datasets.

- **User guidance:** The user study of QUESTO revealed that users often might not know which objectives to specify to improve model performance. Observing participants interact with the system, we realised that they needed system-guided assistance to understand the implication and use-cases of objectives that they can specify. To that end, a VA system may recommend potential objectives that users can specify per iteration, e.g., using notifications or pop-up suggestions. Systems can also use tooltips that explain how a set of objectives can be used to guide the construction of models. Further research can address questions such as: *How can new VA systems be designed that supports user guidance in interactive objective specification?*, *What are the circumstances when a user may not know what objectives to specify or may seem confused about the use case of objectives/constraints?*, *When should a user be notified about objectives that they can consider to specify?* In CACTUS, we supported

users by explicitly showing them conflicts in their objective specification. This guided users to locate the root cause of the problem in the objective function, and to see what they needed to do next to fix the problem. However, guiding users in objective specification still remains an open research challenge that needs further investigation.

- **Managing infeasible results:** Through CACTUS, we learned that in many cases solving conflicting objectives may be mathematically infeasible. Specifically, in the process of modeling, users can specify objectives for which no mathematical solution may exist. In such scenarios, the system may not be able to respond with any solution or may show solutions that perform poorly or are approximate solutions to the desired results. While such solutions are mathematically infeasible or show poor performance, they may be more preferable to users even though their performance is inferior [103]. This was explored in the work of Kamalian et al. where they deployed user interactions in a multi objective optimization problem [114]. Nevertheless, for a better user experience, users may expect to know *why do they see models with low performance or inconsistent output?* (observed through the study of QUESTO, and CACTUS), or *what previous interactions led to the selection of poor models*? In future, VA systems supporting model construction tasks, should educate users about the feasibility of a solution space given a set of specified objectives. This poses a challenge to understand, *how to educate users (without overwhelming them) about mathematically infeasible results given specified preferences?* This space needs further investigation to enlighten approaches to reconcile infeasible user expectations using novel techniques from HCI, information visualization, and visual analytics.

7.2 Reflections

In the following, I summarize my reflection on the two main interactive techniques that I have analyzed through this research.

- **Multi-model steering:** Through this research, I investigated model steering techniques in visual analytics namely, single-model based model steering and multiple model based model steering. In the former, there is a predefined single ML model, while the later uses multiple ML models to help users analyse their data. To that end, I prototyped the VA system Gaggle [225] (with two variants: single and multi-model based) to understand: (1) how a multi-model steering is superior to a single model steering system or vice versa?, and (2) in what circumstances multiple models switch hyperparameters to account for user preferences? I presented the findings from a quantitative and qualitative study highlighting the differences between these approaches. The study showed that a single model system may be adequate for relatively simpler tasks such as binary classification. However, for complex task such as multi-class classification and ranking, where an analysts' definition of class categories may change while they explore data, multi-model based systems are more efficient to help end-users find a preferred ML model.

  Furthermore, in VA systems, model selection is enabled by various techniques which differ by the degree of automation. On one extreme, there is the manual model selection approach, which includes manually choosing a model's learning algorithm and hyperparameter settings from a control panel based visual interface. On the other end is the automated model selection approach (e.g., in systems like Auto-Weka [82],

H2O, Orange-ML, etc.), a technique that automatically selects a model that performs best on an input dataset, specified task, and a chosen performance metric such as precision, recall, etc. In between these two extremes, I explored a semi-automatic model selection approach, where end-users interactively guide the underlying model solver through demonstrated interactions to select a preferred model for their problem. I prototyped two systems namely, BEAMES [50], and Gaggle [225] to validate this approach further. I learned through the controlled-lab user studies that users can create meaningful models by interactively specifying preferences to models instead of resorting to code or using control panel style Auto-ML platforms. Furthermore, the studies confirmed that multiple models better supports users to model their data closer to their expectations. Often in the process of interaction, they could change their task definitions. In such cases multiple models helped to characterize the data more accurately. Finally, I also learned that there is a trade-off between a simpler user experience and granularity of control in adjusting/steering models. While BEAMES allowed users to inspect and steer models with more controls, Gaggle showed a simpler user interface to use with less interaction complexities.

In this thesis, I also presented a year long design study with biology researchers at Georgia Institute of Technology, who cluster genome samples to answer critical analytical questions in their domain. Motivated to test the aforementioned approaches of interactive multi-model steering and model selection, I closely worked with them to build a visual interface that facilitates cluster model recommendations based on user interactions with multiple clustering models (variants of K-means model). To summarize, with this work I validated interactive model selection in a real domain with real data. Through this experience I learned that people prefer to explore data and create models, in the process they tend to test various hypothesis, save different model types to support their hypotheses for further analysis. For this specific work I observed two broad approaches of user interaction, one was top down where users clustered the whole data and then specified their desired cluster memberships interactively. The other approach was bottom up, where users drag dropped data instances from the table to create clusters one by one, and using these interactions, the system learned to find the right hyperparameterization of an optimal clustering model.

- **Interactive objective functions:** While multi-model steering was effective it seemed from the interactions in the other systems, that users can specify a range of implicit feedbacks that often are not communicated back to the user. In some cases users may not understand how each of these preferences affect each other, and more importantly often these preferences when applied together may conflict with one another. In response to these limitations, I presented a novel technique that enables interactive construction of objective functions in classification tasks. Using this approach, a VA system selects models based on user specifications of their preferences in the form of an interactive objective function that is fed to an Auto-ML model solver. Interactive objective function crafted by users is the mathematical embodiment of their requirements (in the form of multiple objectives). This function is utilised to score and rank models that are sampled by the model solver. Based on this ranking, an optimal (preferred) model or a set of top 'k' optimal models are selected for users to inspect. Reflecting back on this work I observed that interactive objective functions employs a feedback loop between the user and the system to communicate intents and

to inspect models through objective specification. Furthermore, I observed that in a multi-objective optimization problem, satisfying every objective might not be feasible, thus formalizing user goals as an objective function facilitates understanding various trade-offs between different models.

As mentioned, multi-objective objective functions may contain objectives that conflict with one another. In this research I developed an interactive algorithmic technique that helps people to: (1) inspect and explore conflicts between objectives, (2) interactively resolve conflicts to construct different variants of the objective function, and (3) incrementally improve the objective function, by comparing its effectiveness in relation to the underlying models' performance (e.g, with respect to metrics such as accuracy score on the validation set). In this work I presented enumeration of potential conflicts between various objectives in interactive objective functions. Grounded on this list of conflicts, I prototyped CACTUS, a VA tool that helps users to specify meaningful objective functions to a classifier by resolving conflicting objectives in their specification. Furthermore, with a quantitative and qualitative user study the I analyzed to find that not only this system provided an expressive visual interface to detect and resolve conflicts but it also motivated users to gamify the process of model creation through designing many variants of objective functions. In future, researchers can investigate novel ways to guide, and encourage participants in this gamification process of model creation. In addition, another area of work that can be extended from this research is how people can collaboratively design objective functions, and then check to see if there are any conflicts between there objective specifications. In practical applications, many ML practitioners form teams to create data science solutions to analytical tasks. In such scenarios, they can collaboratively design objective functions and test its effectiveness using a system like CACTUS.

7.3 Future Work

In the following I list a few potential future research directions that I foresee can result from the contribution of this thesis.

7.3.1 Interaction based user feedback to AutoML with heterogeneous large data sources

In my research I have mostly worked with tabular datasets, to help people build machine learning models using interactive visual interfaces. Recently I have started building applications using text, and time-series data with sequence models (e.g., BERT, CNN) [11] to help domain experts such as urban planners make sense of large-scale social media data [2] (e.g., Twitter and Instagram data). In the future, I seek to explore and investigate interactive human-in-the-loop applications that facilitate constructing models' using diverse heterogeneous large-scale data sources. While this workflow may help to diversify and extend training data sources in interactive model construction processes, it also posits multifarious challenges both in terms of visualizations, interactions, scalability and incorporating user interaction data into ML models. Heterogenous large-scale data sources also pose the problem of data integration, data quality (e.g., noise, missing data), and scalability in relation to interaction and responsiveness in applications. I plan to address these challenges through: (1) the design of new algorithms that help scale multi-model based ML solutions, and (2) the application of longitudinal qualitative and quantitative user research. I aspire

to work on interdisciplinary ML and data problems where heterogeneous data sources can be coupled with user interaction data for large scale applications.

7.3.2 Domain application of human-centered AI modeling

Model selection in machine learning is a known hard problem, even for expert ML practitioners and data scientists. The problem is more pervasive and serious for novices or early adopters in ML. Throughout my research, I sought domain experts' cooperation and collaboration to find access to real-world problems on domain-specific datasets and problem cases. These users may or may not have ML expertise. For example, I have worked with DARPA's explainable AI initiative to build applications to democratize ML solutions for subject matter experts [35, 36] who were novices in ML. Similarly, I have collaborated with urban planning researchers to build an interactive natural language preprocessing (NLP) tool to help build personalized sequence models using AutoML model solvers [53]. In the future, I plan to continue this research and further partner with public agencies or other departments who may need access to ML applications. One such avenue that comes to mind involves healthcare analytics using AI and big data in partnership with medical practitioners. My previous experience of sequence modeling and natural language processing work with urban planners can be extended with state of the art NLP models. In this collaboration, I also plan to research, investigate, and re-define new domain-specific metrics that may be are more meaningful to end-users than conventional ML metrics, such as ROC Curves, Precision, Recall, etc.

7.3.3 Animations in interactive model selection and biases/explainability in ML:

In interactive ML applications, the complexity of ML modeling pipelines often is intimidating. With the advent of recent multiple machine learning based systems, users bear the burden to choose the right model from a large number of choices which can further perplex them. In such cases, users should be eased into the modeling workflow, such that they can better understand the ML processes and make informed decisions as they select models for their analytical tasks. A good example of such a workflow is Google's Teachable Machine Interface [89]. In my research, I have demonstrated numerous interactive algorithmic techniques to simplify the user experience and the user interface to help users interact with multi-model systems [4]. These systems leveraged interactions in visual interfaces, however, lacked to incorporate animations in visual elements (data marks). Research has shown that animation serves as a great asset in narrative visualizations, temporal display of data, and in allowing users to see various phases of a process (e.g., in simulations of genetic programming, fluid modeling, ML optimization of loss functions etc.). Animation can also help to explain complex models such as deep neural networks (e.g., RNN, GPT, BERT etc.) which are hard to reason about in terms of the predictions they make. Animation can help tell users the story of a models' learning process, its latent space representations, and the process of sampling its hyperparameters etc. [55]. As part of future work, I aspire to investigate if animations can be deployed to help users interpret complex ML models and explain human and algorithmic biases in model creation. In this regard, I can foresee that animations in visualization systems can help reduce human and algorithmic biases as people explore data and interactively create models. With the same spirit, I am looking forward to further collaborate with people who have experience with modeling various kinds of biases

in visual analytics. For example, state of the art work in detecting and mitigating biases [241, 242] in visual data analysis can be complemented with animation-based storytelling of model creation processes that emphasize and highlights areas of human and/or algorithmic biases and guide users in ways to mitigate/resolve them.

CHAPTER VIII

CONCLUSION

Table 8: Contributions of this research.

Question		Prototype(s)	Status
RQ1	**What are the various techniques of interactive model construction and selection in human-centered machine learning?**	**BEAMES, Gaggle**	**Published: IEEE CG&A, VDS 2018 [BEST PAPER] Published: Graphics Interface 2020**
RQ2	**How effective are multi-model steering and interactive model selection in supporting domain experts to construct clustering models?**	**Geono-Cluster**	**Published: IEEE TVCG Journal 2020**
RQ3	**What are the interactive techniques that empower people translate their preferences into objective functions?**	**QUESTO**	**Published: Computer Graphics Forum, Eurovis 2020**
RQ4	**How can interactive visual interfaces help users to detect and resolve conflicts in objective functions?**	**CACTUS**	**Under Review: IEEE TVCG Journal 2021**

8.1 Summary

To summarize, my thesis investigated methods to empower users to communicate their preferences to machine learning models using interactive VA systems. I expect my work to help democratize ML to people who may need access to ML-based processes to serve a diverse array of data analysis goals. Through my research, I have answered a set of research questions, as discussed in Chapter 1 (see Table 8). My research contributions can be summarized as below:

1. A set of prototype VA systems that explored various model selection techniques in visual analytics. Furthermore, with a quantitative and qualitative user study showed trade-offs between multi-model steering and single model steering technique in visual analytics.

2. A domain study of interactive model construction and selection technique in visual analytics to support biology researchers cluster genome data.

3. A novel interactive objective function approach that enables users to communicate their preferences and domain knowledge to machine learning models using demonstration based interaction techniques.

4. A quantitative and qualitative study with empirical evidence showing the effectiveness and correctness of interactive objective functions to characterize and translate user preferences to ML models.

5. A visual analytic system that finds and resolves conflicts in objective specification and facilitates exploration of model trade-offs empowering users to select better performing models.

References

[1] "Cars dataset, howpublished = `http://courses.washington.edu/hcde511/s14/datasets/cars.xls`, note = Accessed: 2018-12-11."

[2] "Cities dataset, howpublished = `http://graphics.cs.wisc.edu/vis/explainers/data.html`, note = Accessed: 2018-12-11."

[3] "Movies dataset, howpublished = `https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset#movie_metadata.csv`, note = Accessed: 2018-12-11."

[4] "GWAS Catalog, https://www.ebi.ac.uk/gwas/," 2018.

[5] Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I. J., Hardt, M., and Kim, B., "Sanity checks for saliency maps," in *NeurIPS*, 2018.

[6] Agichtein, E., Brill, E., Dumais, S., and Ragno, R., "Learning user interaction models for predicting web search result preferences," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, (New York, NY, USA), pp. 3–10, ACM, 2006.

[7] Ahamed, F. and Farid, F., "Applying internet of things and machine-learning for personalized healthcare: Issues and challenges," in *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pp. 19–21, Dec 2018.

[8] Akgül, C. B., Rubin, D. L., Napel, S., Beaulieu, C. F., Greenspan, H., and Acar, B., "Content-based image retrieval in radiology: Current status and future directions.," *J. Digital Imaging*, vol. 24, no. 2, pp. 208–222, 2011.

[9] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[10] Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T., "Power to the people: The role of humans in interactive machine learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[11] Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. S., "Effective end-user interaction with machine learning.," in *AAAI*, 2011.

[12] Amershi, S., Fogarty, J., and Weld, D., "Regroup: Interactive machine learning for on-demand group creation in social networks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 21–30, ACM, 2012.

[13] Ancona, M., Ceolini, E., Öztireli, C., and Gross, M., "Towards better understanding of gradient-based attribution methods for deep neural networks," in *ICLR*, 2018.

[14] Andrienko, N., Lammarsch, T., Andrienko, G., Fuchs, G., Keim, D., Miksch, S., and Rind, A., "Viewing visual analytics as model building," *Computer Graphics Forum*, pp. n/a–n/a.

[15] Ankerst, M., Elsen, C., Ester, M., and Kriegel, H.-P., "Visual classification: An interactive approach to decision tree construction," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, (New York, NY, USA), pp. 392–396, ACM, 1999.

[16] Atzmüller, M., Baumeister, J., and Puppe, F., "Introspective subgroup analysis for interactive knowledge refinement," in *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006*, pp. 402–407, 2006.

[17] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., and Suárez, O. D., "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," in *PloS one*, 2015.

[18] Bartholomew, D. J., Steele, F., Galbraith, J., and Moustaki, I., *Analysis of multivariate social science data*. Chapman and Hall/CRC, 2008.

[19] Basu, S., Fisher, D., Drucker, S. M., and Lu, H., "Assisting users with clustering tasks by combining metric learning and classification," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pp. 394–400, AAAI Press, 2010.

[20] Beaudouin-Lafon, M., "Instrumental interaction: An interaction model for designing post-wimp user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, (New York, NY, USA), pp. 446–453, ACM, 2000.

[21] Bell DE, Keeney RL, . R. H., *Conflicting Objectives in Decisions*. John Wiley Sons, 1977.

[22] Bergstra, J., Yamins, D., and Cox, D. D., "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in Science Conference*, pp. 13–20, 2013.

[23] Bernard, J., Zeppelzauer, M., Sedlmair, M., and Aigner, W., "Vial: a unified process for visual interactive labeling," *The Visual Computer*, Mar 2018.

[24] Bouali, F., Guettala, A., and Venturini, G., "Vizassist: An interactive user assistant for visual data mining," *Vis. Comput.*, vol. 32, pp. 1447–1463, Nov. 2016.

[25] Boukhelifa, N., Bezerianos, A., Trelea, I. C., Perrot, N. M., and Lutton, E., "An exploratory study on visual exploration of model simulations by multiple types of experts," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, (New York, NY, USA), pp. 644:1–644:14, ACM, 2019.

[26] Boulianne, S., "Social media use and participation: a meta-analysis of current research," *Information, Communication & Society*, vol. 18, no. 5, pp. 524–538, 2015.

[27] Bradel, L., North, C., House, L., and Leman, S., "Multi-model semantic interaction for text analytics," in *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 163–172, Oct 2014.

[28] Breiman, L., "Bagging predictors," *Mach. Learn.*, vol. 24, pp. 123–140, Aug. 1996.

[29] Brown, E. T., Liu, J., Brodley, C. E., and Chang, R., "Dis-function: Learning distance functions interactively," in *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 83–92, Oct 2012.

[30] Brown, E., Sriram, Y., Cook, K., Chang, R., and Endert, A., "ModelSpace: Visualizing the Trails of Data Models in Visual Analytics Systems," 2018.

[31] Brown, E. T., Liu, J., Brodley, C. E., and Chang, R., "Dis-function: Learning distance functions interactively," in *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, VAST '12, (Washington, DC, USA), pp. 83–92, IEEE Computer Society, 2012.

[32] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G., "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[33] Cao, N., Gotz, D., Sun, J., and Qu, H., "Dicon: Interactive visual analysis of multidimensional clusters," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 2581–2590, Dec 2011.

[34] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N., "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (New York, NY, USA), p. 1721–1730, Association for Computing Machinery, 2015.

[35] Cashman, D., Humayoun, S. R., Heimerl, F., Park, K., Das, S., Thompson, J., Saket, B., Mosca, A., Stasko, J. T., Endert, A., Gleicher, M., and Chang, R., "Visual analytics for automated model discovery," *CoRR*, vol. abs/1809.10782, 2018.

[36] Cashman, D., Xu, S., Das, S., Heimerl, F., Liu, C., Humayoun, S. R., Gleicher, M., Endert, A., and Chang, R., "CAVA: A visual analytics system for exploratory columnar data augmentation using knowledge graphs," *CoRR*, vol. abs/2009.02865, 2020.

[37] Cavallo, M. and Demiralp, , "Clustrophile 2: Guided visual clustering analysis," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[38] Chaofan Chen, Oscar Li, A. B. J. S. C. R., "This looks like that: Deep learning for interpretable image recognition," in *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2019.

[39] Chatzimparmpas, A., Martins, R. M., Kucher, K., and Kerren, A., "Stackgenvis: Alignment of data, algorithms, and models for stacking ensemble learning using performance metrics," 2020.

[40] Chen, J. and Subramanian, L., "Interactive web caching for slow or intermittent networks," in *Proceedings of the 4th Annual Symposium on Computing for Development*, ACM DEV-4 '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[41] Chen, K. and Liu, L., "Vista: Validating and refining clusters via visualization," *Information Visualization*, vol. 3, pp. 257–270, Dec. 2004.

[42] Chen, L., Jose, J. M., Yu, H., and Yuan, F., "A semantic graph-based approach for mining common topics from multiple asynchronous text streams," in *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, (Republic and Canton of Geneva, Switzerland), pp. 1201–1209, International World Wide Web Conferences Steering Committee, 2017.

[43] Chen, M.-L. and Wang, H.-C., "How personal experience and technical knowledge affect using conversational agents," in *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, IUI '18 Companion, (New York, NY, USA), pp. 53:1–53:2, ACM, 2018.

[44] Chen, N.-C., Suh, J., Verwey, J., Ramos, G., Drucker, S., and Simard, P., "Anchorviz: Facilitating classifier error discovery through interactive semantic data exploration," in *IUI '18 23rd International Conference on Intelligent User Interfaces*, pp. 269–280, ACM, March 2018.

[45] Cheng, J. and Bernstein, M. S., "Flock: Hybrid crowd-machine learning classifiers," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, (New York, NY, USA), pp. 600–611, ACM, 2015.

[46] Cleveland, W. S., "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, 1979.

[47] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V., *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[48] Craven, M. W., *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, 1996. AAI9700774.

[49] Crouser, R. J., Franklin, L., Endert, A., and Cook, K., "Toward theoretical techniques for measuring the use of human effort in visual analytic systems," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 121–130, 2017.

[50] Das, Subhajit, C., Dylan, C., Remco, Endert, and Alex, "Beames: Interactive multi-model steering, selection, and inspection for regression tasks," in *IEEE CGA*, 2019.

[51] Das, S., Saket, B., Kwon, B. C., and Endert, A., "Geono-cluster: Interactive visual cluster analysis for biologists," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2020.

[52] Das, S. and Dutt, F., "Inmacs: Interactive modeling and comparison of sentiments from sequence data," Workshop on Data Science with Human in the Loop (DaSH), at KDD 2020, 2020.

[53] Das, S. and Dutt, F., "Inmacs: Interactive modeling and comparison of sentiments from sequence data," *Workshop on Data Science with Human in the Loop (DaSH), at KDD 2020*, 2020.

[54] Das, S., Xu, S., Gleicher, M., Chang, R., and Endert, A., "Questo: Interactive construction of objective functions for classification tasks," *Computer Graphics Forum*, vol. 39, no. 3, pp. 153–165, 2020.

[55] Das Subhajit, Xu Panpan, D. Z. E. A. R. L., "Interpreting deep neural networks through prototype factorization," *ICDM Deep Learning and Clustering Workshop*, Nov 2020.

[56] Datta, S. and Adar, E., "Communitydiff: Visualizing community clustering algorithms," *ACM Trans. Knowl. Discov. Data*, vol. 12, pp. 11:1–11:34, Jan. 2018.

[57] Deb, K. and Saxena, D. K., "On finding pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems,"

[58] Demiralp, Ç., "Clustrophile: A tool for visual clustering analysis," *CoRR*, vol. abs/1710.02173, 2017.

[59] Demšar, J., "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[60] desJardins, M., MacGlashan, J., and Ferraioli, J., "Interactive visual clustering," in *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, (New York, NY, USA), pp. 361–364, ACM, 2007.

[61] Dou, W., Jeong, D. H., Stukes, F., Ribarsky, W., Lipford, H. R., and Chang, R., "Recovering reasoning processes from user interactions," *IEEE Computer Graphics and Applications*, vol. 29, pp. 52–61, May 2009.

[62] Downs, G. M. and Barnard, J. M., "Clustering methods and their uses in computational chemistry," *Reviews in computational chemistry*, vol. 18, pp. 1–40, 2002.

[63] Drori, I., Krishnamurthy, Y., Rampin, R., Lourenço, R., Ono, J. P., Cho, K., Silva, C., and Freire, J., "Alphad3m machine learning pipeline synthesis," 2018.

[64] Drummond, C. and Holte, R. C., "Cost curves: An improved method for visualizing classifier performance," *Mach. Learn.*, vol. 65, pp. 95–130, Oct. 2006.

[65] Du, M., Liu, N., and Hu, X., "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, p. 68–77, Dec. 2019.

[66] Dubey, A., Bhattacharya, I., and Godbole, S., "A cluster-level semi-supervision model for interactive clustering," in *Machine Learning and Knowledge Discovery in Databases* (Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., eds.), (Berlin, Heidelberg), pp. 409–424, Springer Berlin Heidelberg, 2010.

[67] Elmqvist, N., Dragicevic, P., and Fekete, J., "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1539–1148, Nov 2008.

[68] Endert, A., Bradel, L., and North, C., "Beyond control panels: Direct manipulation for visual analytics," *IEEE Computer Graphics and Applications*, vol. 33, pp. 6–13, July 2013.

[69] Endert, A., Han, C., Maiti, D., House, L., Leman, S. C., and North, C., "Observation-level Interaction with Statistical Models for Visual Analytics," in *IEEE VAST*, pp. 121–130, 2011.

[70] Endert, A., Bradel, L., and North, C., "Beyond Control Panels: Direct Manipulation for Visual Analytics," *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 6–13, 2013.

[71] Endert, A., Fiaux, P., and North, C., "Semantic interaction for visual text analytics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), pp. 473–482, ACM, 2012.

[72] Endert, A., Fiaux, P., and North, C., "Semantic interaction for visual text analytics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, (New York, NY, USA), p. 473–482, Association for Computing Machinery, 2012.

[73] Erhan, D., Bengio, Y., Courville, A. C., and Vincent, P., "Visualizing higher-layer features of a deep network," 2009.

[74] Escudero, J., Ifeachor, E., Zajicek, J. P., Green, C., Shearer, J., and Pearson, for the Alzheimer's Disease Neuroimaging Initiative, S., "Machine learning-based method for personalized and cost-effective detection of alzheimer's disease," *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 164–168, Jan 2013.

[75] Fails, J. A. and Olsen, Jr., D. R., "Interactive machine learning," in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, (New York, NY, USA), pp. 39–45, ACM, 2003.

[76] Fails, J. A. and Olsen, Jr., D. R., "Interactive machine learning," in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, (New York, NY, USA), pp. 39–45, ACM, 2003.

[77] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F., "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28* (Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., eds.), pp. 2962–2970, Curran Associates, Inc., 2015.

[78] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F., "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, pp. 2962–2970, 2015.

[79] Fiebrink, R., Cook, P. R., and Trueman, D., "Human model evaluation in interactive supervised learning," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 147–156, ACM, 2011.

[80] Fogarty, J., Ko, A. J., Aung, H. H., Golden, E., Tang, K. P., and Hudson, S. E., "Examining task engagement in sensor-based statistical models of human interruptibility," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, (New York, NY, USA), pp. 331–340, ACM, 2005.

[81] Freund, Y. and Schapire, R. E., "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, (San Francisco, CA, USA), pp. 148–156, Morgan Kaufmann Publishers Inc., 1996.

[82] Garner, S. R., "Weka: The waikato environment for knowledge analysis," in *In Proc. of the New Zealand Computer Science Research Students Conference*, pp. 57–64, 1995.

[83] Gay, G., Rayadurgam, S., and Heimdahl, M. P. E., "Improving the accuracy of oracle verdicts through automated model steering," in *ACM/IEEE International Conference on Automated Software Engineering, ASE '14, Vasteras, Sweden - September 15 - 19, 2014*, pp. 527–538, 2014.

[84] Ghorbani, A., Abid, A., and Zou, J. Y., "Interpretation of neural networks is fragile," in *AAAI*, 2017.

[85] Ghorbani, A., Wexler, J., and Kim, B., "Automating interpretability: Discovering and testing visual concepts learned by neural networks," *ArXiv*, vol. abs/1902.03129, 2019.

[86] Gleicher, M., "A framework for considering comprehensibility in modeling," *Big Data*, vol. 4, Jun 2016. ahead of print.

[87] Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C. D., and Roberts, J. C., "Visual comparison for information visualization," *Information Visualization*, vol. 10, no. 4, pp. 289–309, 2011.

[88] Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D., "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, (New York, NY, USA), p. 1487–1495, Association for Computing Machinery, 2017.

[89] Google, "Google's teachable machines." `https://teachablemachine.withgoogle.com/`. Accessed: 2020-12-11.

[90] Gratzl, S., Lex, A., Gehlenborg, N., Pfister, H., and Streit, M., "Lineup: Visual analysis of multi-attribute rankings," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, vol. 19, no. 12, pp. 2277–2286, 2013.

[91] Guangtao Fu, Zoran Kapelan, Z. K. J. K. J. K. P. R., "Optimal design of water distribution systems using many-objective visual analytics," October 2013.

[92] Guo, D., "Coordinating computational and visual approaches for interactive feature selection and multivariate clustering," *Information Visualization*, vol. 2, pp. 232–246, Dec. 2003.

[93] H., S. G. and S., L. E., "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Human Mental Workload* (H., P. A. and M., N., eds.), vol. 52 of *Advances in Psychology*, pp. 139 – 183, North-Holland, 1988.

[94] Hall, M. A., "Correlation-based feature selection for machine learning," tech. rep., 1999.

[95] Hasan, S. and Ukkusuri, S. V., "Urban activity pattern classification using topic models from online geo-location data," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 363 – 381, 2014.

[96] Hase, P., Chen, C., Li, O., and Rudin, C., "Interpretable image recognition with hierarchical prototypes," *CoRR*, vol. abs/1906.10651, 2019.

[97] He, Z. and Yen, G. G., "Visualization and performance metric in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 386–402, June 2016.

[98] He, Z. and Yen, G. G., "Comparison of visualization approaches in many-objective optimization," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 357–363, June 2017.

[99] Holzinger, A., "Interactive machine learning for health informatics: when do we need the human-in-the-loop?," *Brain Informatics*, vol. 3, pp. 119–131, Jun 2016.

[100] Hu, Y., Milios, E. E., and Blustein, J., "Interactive feature selection for document clustering," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, (New York, NY, USA), pp. 1143–1150, ACM, 2011.

[101] Hughes, E. J., "Evolutionary many-objective optimisation: many once or one many?," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 222–227 Vol.1, 2005.

[102] inc., K., "Kaggle Housing prices Data." `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data/`, 2018. [Online; accessed 15-July-2018].

[103] Jan Hettenhausen, Andrew Lewis, S. M., "Interactive multi-objective particle swarm optimization with heatmap-visualization-based user interface."

[104] Jawaheer, G., Weller, P., and Kostkova, P., "Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback," *ACM Trans. Interact. Intell. Syst.*, vol. 4, pp. 8:1–8:26, June 2014.

[105] Jeong, D. H., Ziemkiewicz, C., Fisher, B., Ribarsky, W., and Chang, R., "ipca: An interactive system for pca-based visual analytics," in *Computer Graphics Forum*, vol. 28, pp. 767–774, Wiley Online Library, 2009.

[106] Jiang, B. and Canny, J., "Interactive machine learning via a gpu-accelerated toolkit," in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, IUI '17, (New York, NY, USA), pp. 535–546, ACM, 2017.

[107] Jianping Zhou, Shawn Konecni, G. G., "Visually comparing multiple partitions of data with applications to clustering," 2009.

[108] Jin, Y. and Sendhoff, B., "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[109] Jin, Y., *Multi-Objective Machine Learning*. Springer, 2006.

[110] Joachims, T., "Optimizing search engines using clickthrough data," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, (New York, NY, USA), pp. 133–142, ACM, 2002.

[111] John Clore, Krzysztof J. Cios, J. D. and Strack, B., "Diabetes 130-us hospitals for years 1999-2008 data set." `https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008`. Accessed : March 15, 2019.

[112] Jr., M. P. P., "Combining classifiers: From the creation of ensembles to the decision fusion," in *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, pp. 1–10, Aug 2011.

[113] Jurek, A., Bi, Y., Wu, S., and Nugent, C., "A survey of commonly used ensemble-based classification techniques," *Knowledge Engineering Review*, vol. 29, no. 5, p. 551–581, 2013.

[114] Kamalian, R., Takagi, H., and Agogino, A. M., "Optimized design of mems by evolutionary multi-objective optimization with interactive evolutionary computation," in *Genetic and Evolutionary Computation – GECCO 2004* (Deb, K., ed.), (Berlin, Heidelberg), pp. 1030–1041, Springer Berlin Heidelberg, 2004.

[115] Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J., "Wrangler: Interactive visual specification of data transformation scripts," in *ACM Human Factors in Computing Systems (CHI)*, 2011.

[116] Kapoor, A., Lee, B., Tan, D., and Horvitz, E., "Interactive optimization for steering machine classification," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 1343–1352, ACM, 2010.

[117] Keim, D. A., Mansmann, F., Schneidewind, J., and Ziegler, H., "Challenges in visual data analysis," in *Proceedings of the Conference on Information Visualization*, IV '06, (Washington, DC, USA), pp. 9–16, IEEE Computer Society, 2006.

[118] Keim, D. A., Mansmann, F., and Thomas, J., "Visual analytics: How much visualization and how much analytics?," *SIGKDD Explor. Newsl.*, vol. 11, pp. 5–8, May 2010.

[119] KERN, LEX, A., GEHLENBORG, N., and JOHNSON, C. R., "Interactive visual exploration and refinement of cluster assignments," *BMC Bioinformatics*, vol. 18, p. 406, apr 2017.

[120] KIM, B., WATTENBERG, M., GILMER, J., CAI, C., WEXLER, J., VIEGAS, F., and SAYRES, R., "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," 2017.

[121] KIM, H., CHOO, J., PARK, H., and ENDERT, A., "Interaxis: Steering scatterplot axes via observation-level interaction," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 1, pp. 131–140, 2016.

[122] KINDERMANS, P.-J., HOOKER, S., ADEBAYO, J., ALBER, M., SCHÜTT, K. T., DÄHNE, S., ERHAN, D., and KIM, B., "The (un)reliability of saliency methods," in *Explainable AI*, 2018.

[123] KLING, F. and POZDNOUKHOV, A., "When a city tells a story: Urban topic analysis," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, (New York, NY, USA), p. 482–485, Association for Computing Machinery, 2012.

[124] KNOWLES, J. and CORNE, D., "Quantifying the effects of objective space dimension in evolutionary multiobjective optimization," in *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, EMO'07, (Berlin, Heidelberg), p. 757–771, Springer-Verlag, 2007.

[125] KOBSA, A., "User modeling in dialog systems: Potentials and hazards," *AI & SOCIETY*, vol. 4, pp. 214–231, Jul 1990.

[126] KOH, P. W. and LIANG, P., "Understanding black-box predictions via influence functions," 2017.

[127] KOMER, B., BERGSTRA, J., and ELIASMITH, C., "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn," in *ICML workshop on AutoML*, 2014.

[128] KOTTHOFF, L., THORNTON, C., HOOS, H. H., HUTTER, F., and LEYTON-BROWN, K., "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *Journal of Machine Learning Research*, vol. 17, pp. 1–5, 2016.

[129] KRAUSE, J., PERER, A., and BERTINI, E., "Infuse: Interactive feature selection for predictive modeling of high dimensional data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 1614–1623, Dec 2014.

[130] KUNCHEVA, L. I., *Combining Pattern Classifiers: Methods and Algorithms*. New York, NY, USA: Wiley-Interscience, 2004.

[131] KWON, B. C., EYSENBACH, B., VERMA, J., NG, K., FILIPPI, C. D., STEWART, W. F., and PERER, A., "Clustervision: Visual supervision of unsupervised clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, pp. 142–151, Jan 2018.

[132] Kwon, B. C., Eysenbach, B., Verma, J., Ng, K., Filippi, C. D., Stewart, W. F., and Perer, A., "Clustervision: Visual supervision of unsupervised clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, pp. 142–151, Jan 2018.

[133] Kwon, B. C., Kim, H., Wall, E., Choo, J., Park, H., and Endert, A., "Axisketcher: Interactive nonlinear axis mapping of visualizations through user drawings.," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 1, pp. 221–230, 2017.

[134] Lee, H., Kihm, J., Choo, J., Stasko, J., and Park, H., "iVisClustering: An Interactive Visual Document Clustering via Topic Modeling," *Computer Graphics Forum*, 2012.

[135] Lee, S. and Koubek, R. J., "Understanding user preferences based on usability and aesthetics before and after actual use," *Interacting with Computers*, vol. 22, no. 6, pp. 530 – 543, 2010. Special Issue on Inclusion and Interaction: Designing Interaction for Inclusive Populations.

[136] Leman, S. C., House, L., Maiti, D., Endert, A., and North, C., "Visual to parametric interaction (v2pi)," *PloS one*, vol. 8, no. 3, p. e50474, 2013.

[137] Letham, B., Rudin, C., McCormick, T. H., and Madigan, D., "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model," 2015.

[138] Lex, A., Schulz, H., Streit, M., Partl, C., and Schmalstieg, D., "Visbricks: Multiform visualization of large, inhomogeneous data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 2291–2300, Dec 2011.

[139] Lex, A., Streit, M., Schulz, H.-J., Partl, C., Schmalstieg, D., Park, P., and Gehlenborg, N., "Stratomex: Visual analysis of large-scale heterogeneous genomics data for cancer subtype characterization," *Comput. Graph. Forum*, vol. 31, pp. 1175–1184, June 2012.

[140] Li, D., Zhang, Y., and Li, C., "Mining public opinion on transportation systems based on social media data," *Sustainability*, vol. 11, no. 15, 2019.

[141] Li, M., Yang, S., and Liu, X., "Diversity comparison of pareto front approximations in many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 44, pp. 2568–2584, Dec 2014.

[142] Li, M., Zhen, L., and Yao, X., "How to read many-objective solution sets in parallel coordinates," *CoRR*, vol. abs/1705.00368, 2017.

[143] Li, O., Liu, H., Chen, C., and Rudin, C., "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *Proceedings of AAAI*, 2018.

[144] Lindroth, P., Patriksson, M., and Strömberg, A.-B., "Approximating the pareto optimal set using a reduced set of objective functions," *European Journal of Operational Research*, vol. 207, no. 3, pp. 1519 – 1534, 2010.

[145] Liu, B., *Sentiment Analysis and Opinion Mining.* Morgan Claypool Publishers, 2012.

[146] Liu, J., Brown, E. T., Chang, R., and Brodley, C. E., "Dis-function: Learning distance functions interactively," in *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, VAST '12, (Washington, DC, USA), pp. 83–92, IEEE Computer Society, 2012.

[147] Liu, S., Wang, X., Liu, M., and Zhu, J., "Towards better analysis of machine learning models: A visual analytics perspective," *Visual Informatics*, vol. 1, no. 1, pp. 48 – 56, 2017.

[148] Lopez-Ornelas, É., Abascal-Mena, R., and Zepeda-Hernández, S., "Social media participation in urban planning: A new way to interact and take decisions," 2017.

[149] Lundberg, S. M. and Lee, S.-I., "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., eds.), pp. 4765–4774, Curran Associates, Inc., 2017.

[150] L'Yi, S., Ko, B., Shin, D., Cho, Y.-J., Lee, J., Kim, B., and Seo, J., "Xclusim: a visual analytics tool for interactively comparing multiple clustering results of bioinformatics data," *BMC Bioinformatics*, vol. 16, p. S5, Aug 2015.

[151] Ma, K., Liao, I., Frazier, J., Hauser, H., and Kostis, H., "Scientific storytelling using visualization," *IEEE Computer Graphics and Applications*, vol. 32, pp. 12–19, Jan 2012.

[152] Malhi, A. and Gao, R. X., "Pca-based feature selection scheme for machine defect classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, pp. 1517–1525, Dec 2004.

[153] Martin, M. E. and Schuurman, N., "Area-based topic modeling and visualization of social media for qualitative gis," *Annals of the American Association of Geographers*, vol. 107, pp. 1028 – 1039, 2017.

[154] Mattoso, M., Dias, J., Ocaña, K. A., Ogasawara, E., Costa, F., Horta, F., Silva, V., and de Oliveira, D., "Dynamic steering of hpc scientific workflows," *Future Gener. Comput. Syst.*, vol. 46, pp. 100–113, May 2015.

[155] Middleton, S. E., De Roure, D. C., and Shadbolt, N. R., "Capturing knowledge of user preferences: Ontologies in recommender systems," in *Proceedings of the 1st International Conference on Knowledge Capture*, K-CAP '01, (New York, NY, USA), pp. 100–107, ACM, 2001.

[156] Ming, Y., Xu, P., Qu, H., and Ren, L., "Interpretable and steerable sequence learning via prototypes," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, (New York, NY, USA), ACM, 2019.

[157] ML, S. L., "Scikit learn model metrics api reference." https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics. Accessed : March 20, 2019.

[158] Monti, S., Tamayo, P., Mesirov, J., and Golub, T., "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Machine Learning*, vol. 52, pp. 91–118, Jul 2003.

[159] Mühlbacher, T., Piringer, H., Gratzl, S., Sedlmair, M., and Streit, M., "Opening the black box: Strategies for increased user involvement in existing algorithm implementations," *IEEE Transactions on Visualization & Computer Graphics*, vol. 20, pp. 1643–1652, Dec 2014.

[160] Mulder, J. D., van Wijk, J. J., and van Liere, R., "A survey of computational steering environments," *Future Gener. Comput. Syst.*, vol. 15, pp. 119–129, Feb. 1999.

[161] Münster, S., Georgi, C., Heijne, K., Klamert, K., Noennig], J. R., Pump, M., Stelzle, B., and [van der Meer], H., "How to involve inhabitants in urban design planning by using digital tools? an overview on a state of the art, key challenges and promising approaches," *Procedia Computer Science*, vol. 112, pp. 2391 – 2405, 2017. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

[162] Nam, E. J., Han, Y., Mueller, K., Zelenyuk, A., and Imre, D., "Clustersculptor: A visual analytics tool for high-dimensional data," in *2007 IEEE Symposium on Visual Analytics Science and Technology*, pp. 75–82, Oct 2007.

[163] Norman, D. A., *Things that make us smart: Defending human attributes in the age of the machine*. Basic Books, 1993.

[164] Nugent, R. and Meila, M., "An overview of clustering applied to molecular biology," in *Statistical methods in molecular biology*, pp. 369–404, Springer, 2010.

[165] Office, T. S. F. C., "Employee compensation data." https://data.sfgov.org/City-Management-and-Ethics/Employee-Compensation/88g8-5mnd. Accessed : March 15, 2019.

[166] Pajer, S., Streit, M., Torsney-Weir, T., Spechtenhauser, F., Möller, T., and Piringer, H., "Weightlifter: Visual weight space exploration for multi-criteria decision making," *IEEE Transactions on Visualization and Computer Graphics*, pp. 611–620, October 2016.

[167] Paparizos, S., Patel, J. M., and Jagadish, H., "Sigopt: Using schema to optimize xml query processing," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 1456–1460, IEEE, 2007.

[168] Patel, K., Bancroft, N., Drucker, S. M., Fogarty, J., Ko, A. J., and Landay, J., "Gestalt: Integrated support for implementation and analysis in machine learning," in *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, (New York, NY, USA), pp. 37–46, ACM, 2010.

[169] Patel, K., Drucker, S., Fogarty, J., Kapoor, A., and Tan, D., "Using multiple models to understand data," pp. 1723–1728, AAAI Press, July 2011.

[170] Patel, K., Fogarty, J., Landay, J. A., and Harrison, B., "Examining difficulties software developers encounter in the adoption of statistical machine learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pp. 1563–1566, AAAI Press, 2008.

[171] Paul, D., Li, F., Teja, M. K., Yu, X., and Frost, R., "Compass: Spatio temporal sentiment analysis of us election what twitter says!," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, (New York, NY, USA), pp. 1585–1594, ACM, 2017.

[172] Pazzani, M. and Billsus, D., "Learning and revising user profiles: The identification of interesting web sites," *Machine Learning*, vol. 27, pp. 313–331, Jun 1997.

[173] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[174] Pelikan, M., Goldberg, D. E., and Cantú-Paz, E., "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, (San Francisco, CA, USA), pp. 525–532, Morgan Kaufmann Publishers Inc., 1999.

[175] Petrie, C. J., Webster, T. A., and Cutkosky, M. R., "Using pareto optimality to coordinate distributed agents," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 9, no. 4, p. 269–281, 1995.

[176] Pezzotti, N., Lelieveldt, B. P. F., van der Maaten, L., Höllt, T., Eisemann, E., and Vilanova, A., "Approximated and user steerable tsne for progressive visual analytics.," *CoRR*, vol. abs/1512.01655, 2015.

[177] Pickles, S., Haines, R., Pinning, R., and Porter, A., "A practical toolkit for computational steering," *Royal Society of London. Proceedings A. Mathematical, Physical and Engineering Sciences*, vol. 363, pp. 1843–1853, 8 2005.

[178] Pike, W. A., Stasko, J., Chang, R., and O'Connell, T. A., "The science of interaction," *Information Visualization*, vol. 8, pp. 263–274, Dec. 2009.

[179] Piringer, H., Berger, W., and Krasser, J., "Hypermoval: Interactive visual validation of regression models for real-time simulation," *Comput. Graph. Forum*, vol. 29, pp. 983–992, 2010.

[180] Plunz, R. A., Zhou, Y., Vintimilla], M. I. C., Mckeown, K., Yu, T., Uguccioni, L., and Sutto, M. P., "Twitter sentiment in new york city parks as measure of well-being," *Landscape and Urban Planning*, vol. 189, pp. 235 – 246, 2019.

[181] Pokharel, R., Haghighi, P. D., Jayaraman, P. P., and Georgakopoulos, D., "Analysing emerging topics across multiple social media platforms," in *Proceedings of*

the Australasian Computer Science Week Multiconference, ACSW 2019, (New York, NY, USA), pp. 16:1–16:9, ACM, 2019.

[182] Potter, K., Wilson, A., Bremer, P. T., Williams, D., Doutriaux, C., Pascucci, V., and Johnson, C. R., "Ensemble-vis: A framework for the statistical visualization of ensemble data," in *2009 IEEE International Conference on Data Mining Workshops*, pp. 233–240, Dec 2009.

[183] Purshouse, R. C. and Fleming, P. J., "On the evolutionary optimization of many conflicting objectives," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

[184] Ragan, E. D., Endert, A., Sanyal, J., and Chen, J., "Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, pp. 31–40, Jan 2016.

[185] Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J., "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *NIPS*, 2017.

[186] Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L., "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, Aug. 2010.

[187] Reed, P. M. and Minsker, B. S., "Striking the balance: Long-term groundwater monitoring design for conflicting objectives," *Journal of Water Resources Planning and Management*, vol. 130, no. 2, pp. 140–149, 2004.

[188] Ren, D., Amershi, S., Lee, B., Suh, J., and Williams, J. D., "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 61–70, Jan 2017.

[189] Ren, D., Höllerer, T., and Yuan, X., "ivisdesigner: Expressive interactive design of information visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 2092–2101, Dec 2014.

[190] Rene Cutura1, Stefan Holzer, M. A. and Sedlmair, M.

[191] Rheingans, P. and desJardins, M., "Visualizing high-dimensional predictive model quality," in *Proceedings of the Conference on Visualization '00*, VIS '00, (Los Alamitos, CA, USA), pp. 493–496, IEEE Computer Society Press, 2000.

[192] Ribeiro, M. T., Singh, S., and Guestrin, C., ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016.

[193] Ribeiro, M. T., Singh, S., and Guestrin, C., ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 1135–1144, ACM, 2016.

[194] Roberts, H., Sadler, J., and Chapman, L., "The value of twitter data for determining the emotional responses of people to urban green spaces: A case study and critical evaluation," *Urban Studies*, vol. 56, no. 4, pp. 818–835, 2019.

[195] Roy, B., "Problems and methods with multiple objective functions," *Mathematical Programming*, vol. 1, pp. 239–266, Dec 1971.

[196] Rudovic, O., Lee, J., Dai, M., Schuller, B., and Picard, R. W., "Personalized machine learning for robot perception of affect and engagement in autism therapy," *Science Robotics*, vol. 3, no. 19, 2018.

[197] Sacha, D., Kraus, M., Bernard, J., Behrisch, M., Schreck, T., Asano, Y., and Keim, D. A., "Somflow: Guided exploratory cluster analysis with self-organizing maps and analytic provenance," *IEEE Transactions on Visualization Computer Graphics*, vol. 24, pp. 120–130, Jan. 2018.

[198] Sacha, D., Sedlmair, M., Zhang, L., Lee, J. A., Peltonen, J., Weiskopf, D., North, S. C., and Keim, D. A., "What you see is what you can change: Human-centered machine learning by interactive visualization," *Neurocomputing*, vol. 268, pp. 164–175, 2017.

[199] Sahu, R. and Chaturvedi, A. K., "Many-objective comparison of twelve grid scheduling heuristics," 2011.

[200] Saket, B., Kim, H., Brown, E. T., and Endert, A., "Visualization by demonstration: An interaction paradigm for visual data exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, pp. 331–340, Jan 2017.

[201] Santos, A. S. R., Castelo, S., Felix, C., Ono, J. P., Yu, B., Hong, S., Silva, C. T., Bertini, E., and Freire, J., "Visus: An interactive system for automatic machine learning model building and curation," *CoRR*, vol. abs/1907.02889, 2019.

[202] Saravanan, R., Asokan, P., and Sachidanandam, M., "A multi-objective genetic algorithm (ga) approach for optimization of surface grinding operations," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 12, pp. 1327 – 1334, 2002.

[203] Sarvghad, A., Saket, B., Endert, A., and Weibel, N., "Embedded merge & split: Visual adjustment of data grouping," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[204] Schneider, B., Jäckle, D., Stoffel, F., Diehl, A., Fuchs, J., and Keim, D. A., "Visual integration of data and model space in ensemble learning," *CoRR*, vol. abs/1710.07322, 2017.

[205] Scitkit-Learn, "Scikit-learn random search hyperparameter tuning."

[206] Sedlmair, M., Heinzl, C., Bruckner, S., Piringer, H., and Möller, T., "Visual parameter space analysis: A conceptual framework," *IEEE Transactions on Visualization & Computer Graphics*, vol. PP, 2014.

[207] Sedlmair, M., Meyer, M., and Munzner, T., "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.

[208] Self, J. Z., Dowling, M., Wenskovitch, J., Crandell, I., Wang, M., House, L., Leman, S., and North, C., "Observation-level and parametric interaction for high-dimensional data analysis," *ACM Trans. Interact. Intell. Syst.*, vol. 8, pp. 15:1–15:36, June 2018.

[209] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D., "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[210] Sener, O. and Koltun, V., "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems 31* (Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., eds.), pp. 527–538, Curran Associates, Inc., 2018.

[211] Seo, J. and Shneiderman, B., "Interactively exploring hierarchical clustering results [gene identification]," *Computer*, vol. 35, pp. 80–86, July 2002.

[212] Seo, J. and Shneiderman, B., "Interactively exploring hierarchical clustering results [gene identification]," *Computer*, vol. 35, pp. 80–86, July 2002.

[213] Settles, B., "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[214] Shang, Z., Zgraggen, E., Buratti, B., Kossmann, F., Eichmann, P., Chung, Y., Binnig, C., Upfal, E., and Kraska, T., "Democratizing data science through interactive curation of ml pipelines," in *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, (New York, NY, USA), pp. 1171–1188, ACM, 2019.

[215] Shashidharan, A., Vatsavai, R. R., Ashish, A., and Meentemeyer, R. K., "tfutures: Computational steering for geosimulations," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, (New York, NY, USA), pp. 27:1–27:10, ACM, 2017.

[216] Shrikumar, A., Greenside, P., and Kundaje, A., "Learning important features through propagating activation differences," *ArXiv*, vol. abs/1704.02685, 2017.

[217] Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M., "Smoothgrad: removing noise by adding noise," 06 2017.

[218] Snoek, J., Larochelle, H., and Adams, R. P., "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25* (Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., eds.), pp. 2951–2959, Curran Associates, Inc., 2012.

[219] Steed, C. A., Ricciuto, D. M., Shipman, G., Smith, B., Thornton, P. E., Wang, D., Shi, X., and Williams, D. N., "Big data visual analytics for exploratory earth system simulation analysis," *Computers Geosciences*, vol. 61, pp. 71 – 82, 2013.

[220] Stiglic, G., Mertik, M., Podgorelec, V., and Kokol, P., "Using visual interpretation of small ensembles in microarray analysis," in *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pp. 691–695, June 2006.

[221] Stolper, C. D., Perer, A., and Gotz, D., "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 1653–1662, Dec 2014.

[222] Stolper, C. D., Perer, A., and Gotz, D., "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 1653–1662, 2014.

[223] Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich, T., Sullivan, E., Drummond, R., and Herlocker, J., "Toward harnessing user feedback for machine learning," in *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, (New York, NY, USA), pp. 82–91, ACM, 2007.

[224] Stumpf, S., Rajaram, V., Li, L., Wong, W.-K., Burnett, M., Dietterich, T., Sullivan, E., and Herlocker, J., "Interacting meaningfully with machine learning systems: Three experiments," *Int. J. Hum.-Comput. Stud.*, vol. 67, pp. 639–662, Aug. 2009.

[225] Subhajit Das, Dylan Cashman, R. C. A. E., "Gaggle: Visual Analytics for Model Space Navigation," in *Graphics Interface*, ACM, 2020.

[226] Sun, Y., Lank, E., and Terry, M., "Label-and-learn: Visualizing the likelihood of machine learning classifier's success during data labeling," in *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, IUI '17, (New York, NY, USA), pp. 523–534, ACM, 2017.

[227] Sundararajan, M., Taly, A., and Yan, Q., "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning* (Precup, D. and Teh, Y. W., eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3319–3328, PMLR, 06–11 Aug 2017.

[228] Talbot, J., Lee, B., Kapoor, A., and Tan, D. S., "Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1283–1292, ACM, 2009.

[229] Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. T., "Adaptively learning the crowd kernel," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, (USA), pp. 673–680, Omnipress, 2011.

[230] Taylor, M. E. and Stone, P., "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Dec. 2009.

[231] Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K., "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 847–855, ACM, 2013.

[232] Tianyi, L., Convertino, G., Wang, W., and Most, H., "Hypertuner: Visual analytics for hyperparameter tuning by professionals," *Machine Learning from User Interaction for Visualization and Analytics, IEEE VIS 2018*, 2018.

[233] Tullio, J., Dey, A. K., Chalecki, J., and Fogarty, J., "How it works: A field study of non-technical users interacting with an intelligent system," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, (New York, NY, USA), pp. 31–40, ACM, 2007.

[234] Urbanek, S., "Exploring statistical forests," 2002.

[235] Ustun, B. and Rudin, C., "Supersparse linear integer models for optimized medical scoring systems," *Machine Learning*, vol. 102, pp. 349–391, 2015.

[236] Van Den Elzen, S. and van Wijk, J. J., "Baobabview: Interactive construction and analysis of decision trees," in *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pp. 151–160, IEEE, 2011.

[237] van Liere, R., Mulder, J. D., and van Wijk, J. J., "Computational steering," *Future Generation Computer Systems*, vol. 12, no. 5, pp. 441 – 450, 1997. HPCN96.

[238] Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L., "Openml: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, 2014.

[239] Walker, D. J., Everson, R., and Fieldsend, J. E., "Visualizing mutually nondominating solution sets in many-objective optimization," *Trans. Evol. Comp*, vol. 17, pp. 165–184, Apr. 2013.

[240] Wall, E., Das, S., Chawla, R., Kalidindi, B., Brown, E. T., and Endert, A., "Podium: Ranking data using mixed-initiative visual analytics," *IEEE Transactions on Visualization Computer Graphics*, vol. 24, pp. 288–297, Jan. 2018.

[241] Wall, E., Blaha, L. M., Franklin, L., and Endert, A., "Warning, bias may occur: A proposed approach to detecting cognitive bias in interactive visual analytics," *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 104–115, 2017.

[242] Wall, E., Blaha, L. M., Paul, C. L., and Endert, A., "A formative study of interactive bias metrics in visual analytics using anchoring bias," in *INTERACT*, 2019.

[243] Wang, X., Zhai, C., Hu, X., and Sproat, R., "Mining correlated bursty topic patterns from coordinated text streams," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), pp. 784–793, ACM, 2007.

[244] Wei, F., Liu, S., Song, Y., Pan, S., Zhou, M. X., Qian, W., Shi, L., Tan, L., and Zhang, Q., "Tiara: A visual exploratory text analytic system," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, (New York, NY, USA), pp. 153–162, ACM, 2010.

[245] Wenskovitch, J. and North, C., "Observation-level interaction with clustering and dimension reduction algorithms," in *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*, HILDA'17, (New York, NY, USA), pp. 14:1–14:6, ACM, 2017.

[246] Wolpert, D. H., "Original contribution: Stacked generalization," *Neural Netw.*, vol. 5, pp. 241–259, Feb. 1992.

[247] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y., "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning* (Bach, F. and Blei, D., eds.), vol. 37 of *Proceedings of Machine Learning Research*, pp. 2048–2057, PMLR, 07–09 Jul 2015.

[248] Yang, Q., Suh, J., Chen, N.-C., and Ramos, G., "Grounding interactive machine learning tool design in how non-experts actually build models," June 2018.

[249] Yeh, C.-K., Kim, B., Arik, S. O., Li, C.-L., Ravikumar, P., and Pfister, T., "On concept-based explanations in deep neural networks," 2019.

[250] Yeh, I. and Lien, C., "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 2473 – 2480, 2009.

[251] Yimam, S. M., Biemann, C., Majnaric, L., Šabanović, Š., and Holzinger, A., "Interactive and iterative annotation for biomedical entity recognition," in *Brain Informatics and Health* (Guo, Y., Friston, K., Aldo, F., Hill, S., and Peng, H., eds.), (Cham), pp. 347–357, Springer International Publishing, 2015.

[252] Zeiler, M. D. and Fergus, R., "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[253] Zhang, L., Madigan, C. F., Moskewicz, M. H., and Malik, S., "Efficient conflict driven learning in a boolean satisfiability solver," in *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '01, p. 279–285, IEEE Press, 2001.

[254] Zhang, S. and Feick, R., "Understanding public opinions from geosocial media," *ISPRS International Journal of Geo-Information*, vol. 5, no. 6, 2016.

[255] Zhang, Y., Wang, D., and Li, T., "idvs: An interactive multi-document visual summarization system," in *ECML/PKDD*, 2011.

[256] Zhou, B., Sun, Y., Bau, D., and Torralba, A., "Interpretable basis decomposition for visual explanation," in *ECCV*, 2018.

[257] Zhou, Y. and Qiu, G., "Random forest for label ranking," *CoRR*, vol. abs/1608.07710, 2016.

[258] Zhu, X., Singla, A., Zilles, S., and Rafferty, A. N., "An overview of machine teaching," *CoRR*, vol. abs/1801.05927, 2018.